Titre: Title:	Intégration et optimisation multidisciplinaire d'une turbine à gaz au stade préliminaire de conception
Auteur: Author:	Jasmin Turcotte
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Turcotte, J. (2004). Intégration et optimisation multidisciplinaire d'une turbine à gaz au stade préliminaire de conception [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7304/

UNIVERSITÉ DE MONTRÉAL

INTÉGRATION ET OPTIMISATION MULTIDISCIPLINAIRE D'UNE TURBINE À GAZ AU STADE PRÉLIMINAIRE DE CONCEPTION

JASMIN TURCOTTE
DÉPARTEMENT DE GÉNIE MÉCANIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE MÉCANIQUE)
JANVIER 2004

Your file Votre référence
ISBN: 0-612-90863-1
Our file Notre référence
ISBN: 0-612-90863-1

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

INTÉGRATION ET OPTIMISATION MULTIDISCIPLINAIRE D'UNE TURBINE À GAZ AU STADE PRÉLIMINAIRE DE CONCEPTION

présenté par : TURCOTTE Jasmin

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. CAMARERO Ricardo, Ph.D., président

M. TRÉPANIER Jean-Yves, Ph.D., membre et directeur de recherche

M. DION Michel, B.Eng., membre et codirecteur de recherche

M. FORTIN Clément, Ph.D., membre

À tout ceux qui m'ont supporté
durant mes études.

REMERCIEMENTS

Merci d'abord à M. Jean-Yves Trépanier et à M. Michel Dion, directeur et codirecteur respectivement et instigateurs de ce projet de maîtrise. Vos conseils et votre disponibilité m'ont beaucoup aidé lors de la réalisation ce projet. Ce fut une expérience très enrichissante pour moi de réaliser un projet en collaboration avec P&WC sous votre direction.

Merci à Christophe Tribes pour son implication dans la partie optimisation de ce projet. Tes précieux conseils ont vraiment contribué au succès de ce projet.

Merci à M^{me} Yulia Panchenko, M. Michel Dion, M. Ghislain Plante et M. John Wheeler, tous ingénieurs chez P&WC, pour leur support technique et leur grande disponibilité. Ce fut enrichissant et très agréable de vous côtoyer tout au long de ma maîtrise.

Merci au FQRNT (anciennement le FCAR) pour la bourse de maîtrise recherche. Ce support financier m'a permis de réaliser mon projet tout en maintenant une bonne qualité de vie.

Merci finalement à P&WC d'avoir financé et fourni les locaux et l'équipement nécessaire à la réalisation de ce projet.

RÉSUMÉ

Ce mémoire présente l'intégration et l'optimisation multidisciplinaire d'une turbine à gaz au stade préliminaire de conception. Les trois disciplines suivantes sont prises en compte : performances, aérodynamique et structures rotatives. L'intégration est effectuée par l'utilisation d'enveloppes (« wrappers » en anglais) écrites en langage Perl et servant à envelopper les programmes d'analyse associés à chaque discipline.

L'optimisation est effectuée dans le logiciel iSIGHT avec un algorithme d'optimisation non linéaire de type « Sequential Quadratic Programming » (SQP) nommé NLPQL.

Les disciplines étudiées dans ce travail sont couplées ce qui implique la résolution d'un système multidisciplinaire (MDA pour « Multi-Disciplinary Analysis ») afin d'obtenir des résultats. Deux formulations à un niveau permettant de gérer la résolution du problème MDA sont étudiées afin de comparer leur performances. La première, nommée « Fully Integrated Optimization » (FIO), réalise l'optimisation de la turbine en résolvant le problème MDA à chaque fois que l'optimiseur requiert une analyse de la turbine. La seconde, nommée « Distributed Analysis Optimization » (DAO), découple le problème MDA par l'introduction de contraintes additionnelles ce qui permet à l'optimiseur de s'affranchir de la résolution du problème MDA à chaque analyse de la turbine.

L'étude démontre le potentiel de la stratégie DAO à fournir une solution optimale en beaucoup moins d'évaluations et de temps de calcul que FIO pour un problème légèrement couplé avec beaucoup de variables et de contraintes. On démontre également que DAO a le potentiel de fournir une solution convergée en environ une minute si le calcul des gradients est parallélisé.

ABSTRACT

This master's thesis present the integration and Multidisciplinary Design Optimization (MDO) of a gas turbine problem using three disciplines : performance, aerodynamics and rotating structures. The integration is done using wrappers written in Perl language.

The MDO is conducted using the iSIGHT software and a Sequential Quadratic Programming method (SQP) named NLPQL.

The resolution of Multi-Disciplinary Analysis (MDA) problem has to be done in this work because there is a coupling between the disciplines studied. Two single level strategies are studied to resolve the MDA problem, in order to compare their performances. The first one, named Fully Integrated Optimization (FIO), solve the MDA problem each time the optimizer need a turbine analysis. The second one, named Distributed Analysis Optimization (DAO), introduce additionnal constraints to relax the coupling and avoid the need to solve the MDA problem at each turbine analysis.

Results show the potential of the DAO strategy to give a converged optimal solution in less evaluation calls and execution time then the FIO strategy. Moreover, it can be seen that a converged DAO optimization solution could be obtained in less than a minute if gradients computation were parallelized on a distributed network.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES SIGLES ET ABRÉVIATIONS

Sigles

FCAR	Fonds pour la Formation de Chercheurs et l'Aide à la Recherche
FQRNT	Fonds Québecois de Recherche sur la Nature et les Technologies
P&WA	Pratt & Whitney America
P&WC	Pratt & Whitney Canada
UTC	United Technology Corporation

Abréviations

BT	« Block Test »
DAO	Distributed Analysis Optimisation
DOE	Design Of Experiment
FIO	Fully Integrated Optimisation
HPT	Turbine à haute pression (High Pressure Turbine)
KKT	Karush-Kuhn-Tucker
MDA	Multi-disciplinary Design Analysis
MDO	Multi-disciplinary Design Optimisation
Mdol	Multi-disciplinary Optimisation Language
PMDO	Preliminary Multidisciplinary Design Optimisation
PT	Turbine de puissance (Power Turbine)
P1242	Programme « mean line » du département d'aérodynamique
RD	RotorDesigner
SF	Facteur d'échelle (« Scale Factor »)
sfc	Consommation spécifique « Specific Fuel Consumption »
SOAPP	State-Of-the-Art Performance Program

Tcl	Tool Command Language
TDF	Turbine Data File

Notations

N	nombre de discipline
F_i	fonction coût associée à la discipline i
C_i	vecteur des contraintes associées à la discipline i
s	vecteur des variables communes à deux disciplines ou plus
l_i	vecteur des variables propres à la discipline i
p_i	vecteur des paramètres de la discipline i
r_i	vecteur des résultats de la discipline i
t_i	vecteur des variables de relaxation du couplage de la discipline i
$\tilde{a}$	valeur adimensionnelle de a

Indices

v	Aubes (« vane »)
b	Ailettes (« blade »)
x	Distance axiale
t	À l'extrémité (« tip »)
h	À l'emplanture (« hub »)
i	$\{SOAPP, P1242, RD\}$ ou {étage 1, étage 2}
C	Valeur thermodynamique corrigée (adimensionnalisée)
0	Entrée du moteur ou de la turbine
3	Entrée du brûleur
4	Entrée de la turbine
56	Section entre 2 étages de turbine

LISTE DES ANNEXES

AVANT-PROPOS

Ce sujet de recherche a été choisi pour les défis qu'il comporte, tant au niveau de l'intégration que de l'optimisation. Il retient l'attention des industries aérospatiales tel que P&WC depuis plusieurs années. Dans le passé, les tentatives de PMDO chez P&WC et P&WA se sont soldées par des succès partiels pour ce type de problème. Le présent mémoire vise à faire l'étude d'un cas quelque peu simplifié afin d'identifier plus facilement les difficultés et de trouver un moyen de les surmonter. L'étude d'un cas complet est déjà en préparation chez P&WC et ce mémoire pourra donc servir, en partie, de balise pour les ingénieurs et les étudiants qui seront appelés à travailler sur ce projet d'envergure.

INTRODUCTION

La concurrence dans le domaine de l'aéronautique a poussé Pratt & Whitney Canada (P&WC) et bien d'autres compagnies à vouloir sans cesse diminuer leur coût de développement. Un des moyens d'y parvenir est d'écourter le temps de design au maximum. C'est dans ce contexte que P&WC a démarré en 2000 un projet nommé PMDO [1], pour « Preliminary Multidisciplinary Design Optimization ». Le projet PMDO a comme objectif d'accélérer au maximum le design préliminaire tout en l'optimisant. Il va sans dire que ceci est très important car il est admis que le design préliminaire a une influence déterminante sur le produit fini. À l'heure actuelle, cette étape est dans bien des cas faite d'une manière imparfaite par rapport à son influence sur le design en entier et ceci force les ingénieurs à utiliser des données incomplètes, en plus d'augmenter le risque d'erreur. De plus, la complexité du problème fait en sorte que des être humains ne peuvent pas tout prendre en compte et ceci peut mener à des designs préliminaires partiellement réalisables et non-optimaux. Le projet PMDO vise donc à prendre en compte toutes les disciplines tout en ayant accès à toutes les données pour être en mesure de fournir un design préliminaire complètement réalisable et optimal. Finalement, le projet a le potentiel d'éliminer les sources d'erreurs humaines et de réaliser le design plus rapidement grâce à un optimiseur qui exécute les programmes et recherche le design optimal. C'est dans ce contexte de l'optimisation multidisciplinaire que s'inscrit le présent mémoire de maîtrise.

Les objectifs du projet

Le contexte de l'optimisation multidisciplinaire apporte plusieurs défis, incluant le choix d'une stratégie d'intégration, le choix d'une méthode d'optimisation et le choix d'une stratégie pour l'optimisation des problèmes couplés.

Le premier objectif de ce projet de maîtrise est d'explorer et de comparer la performance de deux stratégies pour l'optimisation des systèmes couplées. La première stratégie, nommée « Distributed Analysis Optimization » (DAO) et la seconde stratégie, appelée « Fully Integrated Optimization » (FIO), seront comparées pour un problème d'optimisation multidisciplinaire avec des programmes couplés ayant beaucoup de variables et de contraintes pour des applications réelles, utilisées en industrie.

Pour ce faire, l'optimisation d'une turbine à gaz fictive définie chez P&WC sert de modèle. L'objectif des ingénieurs chez P&WC est d'obtenir rapidement un design préliminaire optimal par l'utilisation d'un optimiseur. C'est dans ce contexte que les stratégies DAO et FIO seront testées avec un optimiseur de type SQP nommé NLPQL. Bien que les optimiseurs de type SQP soient réputés pour leur efficacité à trouver un minimum, cette classe d'optimiseur ne garantit pas de trouver un unique minimum dans le cas où la fonction à minimiser est non-linéaire et présente de nombreux minimums locaux. Un second objectif du mémoire consiste à valider ce choix de méthodes de type SQP. Le projet est réalisé à l'aide de l'environnement d'optimisation nommé iSIGHT déjà en utilisation chez P&WC.

Finalement, un troisième objectif, connexe mais important, est d'explorer une méthode d'intégration des programmes d'analyse basée sur une interaction entre iSIGHT et le langage Perl. Ceci permettra de comparer les performances de ce type d'intégration avec une intégration complètement réalisée à l'aide du langage Mdol, propre à iSIGHT, dans un projet antérieur chez P&WC.

Le système PMDO développé pour le présent projet se caractérise donc principalement par l'utilisation conjointe du langage Perl[1] et du logiciel d'optimisation

[1]http ://www.cpan.org

iSIGHT[2]. Le logiciel iSIGHT propose un environnement intégré et convivial pour l'intégration des processus liés à l'optimisation. Il propose une variété d'optimiseurs et P&WC utilise cet environnement pour tout ce qui touche à l'optimisation. Le langage Perl a été choisi pour réaliser l'enrobage des programmes car il est idéal pour ce type de tâches, c'est-à-dire lire et écrire dans des fichiers textes.

Pour ce qui est du problème de design en tant que tel, le nombre de disciplines impliquées dans le processus de design préliminaire d'une turbine à gaz dépasse la douzaine. Dans le but de limiter l'ampleur du présent travail tout en permettant de disposer d'un problème représentatif, l'intégration des trois programmes principaux contrôlant largement les performances de la turbine a été réalisée dans le présent travail. Le premier programme, appelé SOAPP, analyse le cycle thermodynamique et fournit les valeurs des températures et pressions à différentes stations de la turbine. Le deuxième, appelé P1242, analyse l'écoulement et fournit les rendements de chaque turbine, les triangles de vitesse, les pertes de charges, etc. Le troisième, appelé Rotor Designer (RD), analyse les contraintes dans les structures rotatives et donne la durée de vie de la turbine, la marge de résistance à la survitesse, les contraintes dans les pièces, etc. Ces programmes sont décrits plus en détail au chapitre 2. Par la suite, des optimisations doivent être conduites en utilisant iSIGHT avec pour but de trouver le plus rapidement possible une solution optimale, c'est-à-dire la solution qui minimisera la consommation spécifique de carburant, ou *sfc*. Actuellement, chacun de ces trois programmes est la responsabilité d'un département chez P&WC et l'optimisation est faite par les ingénieurs qui s'échangent de l'information pendant quelques cycles, jusqu'à ce qu'un design réalisable soit obtenu. Cette manière de faire est souvent longue et vu le nombre élevé de variables couplées et partagées, il n'est pas du tout évident que les solutions trouvées par les ingénieurs soient optimales.

[2]http ://www.engineous.com

Approche envisagée

La méthodologie se divise en deux volets principaux. Le premier concerne l'intégration des différents programmes de design préliminaire. La création d'un fichier commun contenant la valeur courante des variables pour chaque programme permettra de centraliser l'information critique. Les divers programmes seront enrobés à l'aide du langage Perl. L'avantage premier est qu'iSIGHT lira et écrira dans un seul fichier minimisant ainsi la dépendance du système envers iSIGHT.

Le second volet du projet concerne l'optimisation. La méthode à gradient nommée NLPQL est utilisée pour conduire les optimisations car ce type de méthodes convient bien aux stratégies FIO et DAO. De plus, ces méthodes sont en générale très efficace à trouver un optimum. Par contre, dans le problème qui nous intéresse, rien ne garantit que l'optimum trouvé est un optimum global, tel que mentionné précédemment. Il demeure que le point de départ étant défini par des ingénieurs d'expérience, l'optimiseur a donc de bonnes chances d'être démarré à un endroit convenable dans l'espace de design, facilitant ainsi sa convergence.

Dans l'ordre chronologique, les opérations suivantes ont été effectuées lors de ce projet :

1. Définition du problème (chapitre 2)
2. Paramétrisation (annexe I)
3. Intégration des programmes selon la paramétrisation (chapitre 4)
4. Décomposition du problème selon FIO et DAO (chapitre 3)
5. Adimentionalisation du problème (chapitre 2)
6. Intégration du problème dans iSIGHT (chapitre 4)
7. Réduction du problème (chapitre 2)
8. Optimisation (chapitre 3)

9. Résultats et analyse (chapitre 5)

La présentation de ce mémoire suivra plutôt la structure suivante : Au chapitre 1, une revue de littérature permettra de situer le présent travail en relation avec les travaux publiés dans la littérature. Le chapitre 2 détaillera la modélisation du problème MDO à optimiser. Le chapitre 3 décrira en détail comment le problème MDO a été implanté à l'aide des stratégies FIO et DAO. Le chapitre 4 fournira l'information relative à l'intégration. Finalement, le chapitre 5 présentera les résultats et une discussion appropriée.

Notes

Il est important de noter que toutes les références à des temps de calcul, ou autres ressources informatiques, correspondent à une plateforme HP C-3600 avec le système d'exploitation HP-UX 10.0.

Certains termes utilisés dans ce mémoire sont en anglais. Ils sont pour la plupart des noms de variables tel qu'utilisés à l'interne chez P&WC. Ils n'ont pas été traduits car pour la majorité d'entre eux, l'équivalent français n'est jamais utilisé ou n'existe pas. On s'assure ainsi que les lecteurs de P&WC pourront comprendre parfaitement le travail réalisé.

CHAPITRE 1

REVUE BIBLIOGRAPHIQUE

Dans le présent chapitre, on présente des articles de la littérature relatifs aux aspects théoriques de l'optimisation et de la décomposition afin de mettre le lecteur en contexte. Des exemples d'applications vont ensuite permettre de comparer les différentes méthodes de décomposition. La revue bibliographique vient ainsi appuyer l'objectif principal de ce mémoire qui est de comparer les performances des stratégies FIO et DAO pour un problème multidisciplinaire avec un grand nombre de variables et de contraintes et des fonctions provenant de programmes réels qui s'exécutent rapidement.

1.1 Optimisation et décomposition

Les principales raisons pour décomposer un problème d'optimisation multidisciplinaire sont le couplage entre les différentes disciplines et la répartition de la complexité du problème initial dans plusieurs sous-problème plus petits et plus facile à résoudre. On parle de couplage lorsque les sorties d'un programme sont les entrées d'un autre, tel qu'expliqué par Alexandrov [2]. Un exemple souvent retrouvé dans la littérature est celui de l'interaction entre un fluide et une structure comme une aile d'avion [3, 4, 5, 6, 7], une pale d'hélicoptère [8] ou une ailette de turbine [9].

Il est possible d'éviter le problème du couplage en faisant une optimisation séquentielle mais comme l'ont montré Korte *et al.* [3], une optimisation multidisciplinaire donne de meilleurs résultats. Une optimisation séquentielle consiste à optimiser une discipline à la fois, par rapport aux variables qui dépendent de cette discipline, plutôt que d'optimiser avec toutes les disciplines simultanément. Korte a comparé l'approche FIO avec une optimisation séquentielle d'un « aerospike nozzle ». Le problème faisait

intervenir un léger couplage entre l'aérodynamique et la structure. La conclusion de Korte est qu'il est préférable de gérer le couplage par une méthode de décomposition plutôt que par une méthode d'optimisation séquentielle.

On peut diviser les méthodes de décomposition selon les deux catégories suivantes : les méthodes à un niveau et les méthodes multi-niveaux. Dans les méthodes à un niveau, il n'y a qu'un seul optimiseur qui contrôle l'ensemble des variables tandis que dans les méthodes multi-niveaux, il y a l'optimiseur du problème maître qui contrôle une partie des variables et qui interagit avec des sous-optimiseurs contrôlant chacun un sous-problème. Le problème d'optimisation reste donc entier dans les méthodes à un niveau et n'est par conséquent pas décomposé. On parle tout de même de méthode de décomposition à un niveau dans la littérature. Les méthodes à un niveau permettent toutefois la décomposition des différentes analyses et on parle alors de découplage du problème MDA (« Multidisciplinary Analysis »). On compte plusieurs stratégies permettant de découpler ou non le problème MDA et appartenant à la famille des méthodes à un niveau. Il y a notamment, les stratégies « Fully Integrated Optimization » (FIO) et « Distributed Analysis Optimization » (DAO) étudiées dans le présent travail. Parmi les méthodes multi-niveaux, on compte « Collaborative Optimization » (CO) et « Optimization by Linear Decomposition » (OLD). Toutes ces méthodes ont déjà été présentées, notamment par Braun [10], Tribes [11], Alexandrov [2, 12, 13, 14], Cramer [15] et Röhl [16].

Alexandrov et Lewis [12] ont démontré mathématiquement que les méthodes multi-niveaux comme CO et OLD sont potentiellement inefficaces lorsque qu'elles sont utilisées avec des algorithmes d'optimisations non linéaires conventionnels comme les méthodes SLP et SQP (« Sequential [Linear,Quadratic] Programming »). Ils ont de plus démontré [13, 17] que les formulations FIO et DAO sont équivalentes, donc qu'elles ont les mêmes propriétés de convergence. Ceci est important car on a ainsi l'assurance

mathématique que si la formulation FIO initiale est faisable, alors son équivalent DAO l'est aussi. De plus, si on combine la méthode DAO avec un optimiseur qui gère bien les contraintes d'égalités, elle est sensée être efficace car ses propriétés de convergence sont les mêmes que FIO. Cette preuve implique donc que si on veut utiliser un algorithme d'optimisation conventionnel, il vaut mieux utiliser une méthode de décomposition à un niveau comme FIO ou DAO afin de minimiser les sources de problèmes.

Les méthodes SLP et SQP consistent à approximer localement l'espace de design respectivement de façon linéaire et quadratique. Parmi les méthodes SQP les plus reconnues et éprouvées, on retrouve DONLP écrit par Spellucci [18] et NLPQL écrit par Schittkowski [19]. Ces méthodes sont accessibles via le logiciel d'optimisation iSIGHT.

1.2 Applications comparant les méthodes de décomposition

Kroo [6] a comparé les méthodes FIO et DAO pour le design préliminaire d'un avion. Il a étudié un problème faisant intervenir 3 disciplines sévèrement couplées et comportant au total 13 variables et 5 contraintes pour la formulation FIO et 18 variables et 14 contraintes pour la formulation DAO. Un couplage est sévère si sa résolution demande un grand nombre d'itérations. La conclusion de Kroo est que DAO réduit de beaucoup le temps de calcul et le nombre d'évaluation des programmes d'analyse par rapport à FIO pour un problème avec peu de variables et de contraintes, un couplage sévère et des programmes rapides. L'optimisation fut effectuées avec NPSOL qui est un optimiseur conventionnel de type SQP.

Alexandrov et Kodiyalam [20, 21] ont comparé FIO, DAO et CO sur 5 problèmes d'optimisation différents. Dans tous les cas, il n'y avait jamais plus d'une dizaine de variables et de contraintes avec la formulation FIO. Pour DAO et CO, quelques

variables sont ajoutées pour gérer les couplages mais leur nombre ne dépasse jamais la trentaine. Les programmes d'analyse utilisés étaient tous très rapides. Les optimisations avec les 3 stratégies de décomposition ont été effectuées avec des méthodes d'optimisation conventionnelles de type SLP ou SQP. Chaque optimisation a été menée jusqu'à ce que le meilleur optimum connu soit atteint, si possible, en partant de plusieurs points différents. Le couplage entre les disciplines a requis, en moyenne, entre 1 itération (pour le cas 1) et 16 itérations (pour le cas 5). Le résultat final est que CO est toujours moins efficace que FIO et DAO en terme du nombre d'évaluations et de la capacité à atteindre l'objectif. Pour les deux premiers cas, FIO est nettement supérieur à DAO et pour les 3 autres, le contraire a été observé. FIO et DAO réussissent presque toujours à atteindre l'objectif pour tous les points de départ à l'exception d'un cas où FIO a échoué 7 fois en 12 tentatives. On tire pour conclusion que DAO semble supérieur à FIO, en terme du nombre d'évaluation, lorsque le couplage entre les disciplines est difficile à résoudre. Il semble que la capacité de DAO de relaxer le couplage pendant l'optimisation est justifiée.

1.3 Résumé

En résumé, Korte [3] a montré qu'il est plus avantageux de décomposer un problème MDO plutôt que de faire une optimisation séquentielle. Alexandrov et Lewis [12] ont ensuite démontré que la décomposition d'un problème MDO à un niveau est préférable à une décomposition multi-niveau lorsqu'une méthode conventionnelle d'optimisation est utilisée. Kroo [6] présente un exemple d'application dans lequel il démontre la supériorité de DAO sur FIO pour un problème avec peu de variables et de contraintes, un couplage sévère et des programmes d'analyse rapides. Finalement, Alexandrov et Kodiyalam [20, 21] présentent 5 cas tests avec des couplages de différentes sévérités avec pour conclusion que DAO a nettement l'avantage sur FIO pour des couplages sévères.

La théorie et les applications semblent donc montrer que pour des problèmes cou-

plés de petites tailles, il vaut mieux d'une part utiliser une méthode de décomposition à un niveau si on veut optimiser avec un optimiseur conventionel et d'autre part que si le couplage entre les disciplines est sévère, DAO est supérieur à FIO. Le présent travail vise à comparer FIO et DAO sur un problème de grande taille avec un couplage plus léger explorant ainsi le comportement des deux méthodes pour une autre classe de problème.

CHAPITRE 2

MODÉLISATION DU PROBLÈME

La modélisation complète d'un problème de PMDO se fait en plusieurs étapes. Le but est d'obtenir un modèle mathématique paramétrique complet de la turbine à gaz. On commence donc par présenter le problème physique et les modèles disciplinaires utilisés. On paramétrise ensuite le problème afin d'obtenir un modèle mathématique paramétrique de la turbine qui sera ensuite raffiné par une adimensionnalisation ainsi qu'une réduction du nombre de variables et de contraintes.

La paramétrisation est très technique et est en grande partie déterminée par les entrées requises par les programmes utilisés dans cette étude. Elle a été entièrement définie par les ingénieurs et n'a donc pas fait l'objet de recherche lors de ce projet. On la présente donc à l'annexe I.

2.1 Définition physique du problème

La première étape de la modélisation consiste à identifier les différentes disciplines et applications liées à l'analyse de la turbine. Dans ce projet, trois disciplines sont prises en considération : la performance, l'aérodynamique et les structures rotatives. On choisit de se limiter uniquement à ces trois disciplines pour trois raisons principales :

1. L'aspect thermodynamique, l'aérodynamique et les structures rotatives influencent le plus fortement les caractéristiques finales d'une turbine à gaz ;
2. On veut étudier un cas simple mais représentatif de ce que les ingénieurs font actuellement comme design préliminaire ;
3. Il n'y a pas d'outils de design préliminaire adéquats pour les autres disciplines

(structures statiques, air de refroidissement, coût, poids, etc). On entend par adéquat : robuste, rapide et précis.

Les trois disciplines, performance, aérodynamique et structure rotative, font intervenir les quatre applications suivantes : SOAPP, P1242, RD et P0831. Les sous-sections qui suivent présentent ces différents programmes. Une définition sommaire de la nomenclature d'une turbine à gaz est ensuite présentée. Finalement, la turbine à gaz utilisée durant ce projet est décrite.

2.1.1 Description de SOAPP

SOAPP (State-Of-the-Art-Performance-Program) est le programme de performance de P&WC. C'est un programme 0-D, c'est-à-dire que les différentes stations de la turbine à gaz sont représentées dans le modèle comme des noeuds reliés entre eux par des transformations thermodynamiques. John A. Reed [22, 23], Kim *et al.* [24], Eveker [25] et Kurzke [26] sont d'excellentes références concernant des programmes de performance du même type que SOAPP.

SOAPP fournit l'ensemble des données du cycle thermodynamique du moteur, incluant la consommation spécifique de carburant (*sfc*), la puissance, les températures et pressions. Les rendements des différents étages de turbine sont des entrées de SOAPP et seront obtenus de P1242, le programme d'aérodynamique. SOAPP s'exécute en environ une seconde pour le moteur étudié. Il est stable mais peut tout de même échouer si les conditions d'entrée ne font pas de sens physique.

L'interface de SOAPP est un langage de programmation qui permet de générer un programme spécifique à chaque moteur. Aucun format de référence n'existe chez P&WC afin de générer des programmes en SOAPP. Donc, le format des fichiers d'entrée et de sortie dépend de l'ingénieur qui a créé le modèle SOAPP pour un moteur

en particulier. Ceci vient grandement limiter le nombre de cas tests qu'on peut étudier dans ce projet. En effet, il est quasiment impossible de créer un « parser » universel pour SOAPP puisqu'il n'y a pas de format standard. Il faudrait qu'un format standard pour le fichier d'entrée de SOAPP soit fixé pour qu'un parser universel soit réalisable. On limite donc la présente étude à un seul moteur et le « parser » en Perl pour ce dernier sera créé. Puisque le but de ce projet est d'avantage centré sur l'optimisation et la décomposition plutôt que l'intégration générique, l'étude se limitera à un seul moteur.

2.1.2 Description de P1242

P1242 est un programme d'analyse de l'écoulement des gaz dans la turbine. C'est un programme 1-D et demi ou « mean line ». Le principe d'un « mean line » est d'approximer l'écoulement dans la turbine par celui à la ligne radiale moyenne du moteur. On peut ensuite calculer les propriétés de l'écoulement à l'aide de formules analytiques et empiriques. Les références pour ce type d'application sont nombreuses et les articles de J.D. Denton [27] et de Kacker et Okappu [28] sont particulièrement pertinents. P1242 est un programme 1-D et demi car il tient compte des dimensions radiales (aires de passage) dans les calculs. Il fournit ainsi principalement les distributions radiales et axiales de pression et de température, le rendement de la turbine et de chaque étage de turbine, les pertes de charges, les angles de l'écoulement et les nombres de Mach. La version utilisée dans ce projet inclue également des calculs d'air de refroidissement. P1242 s'exécute en environ 0.3 seconde pour le moteur étudié.

P1242 est plutôt stable. Il peut tout de même échouer si la solution est irréalisable ou en dehors des corrélations. Une solution est irréalisable quand, par exemple, on demande à la turbine de fournir plus de travail qu'elle ne peut physiquement en produire pour les conditions d'opération et la géométrie données. Pour ce qui est des analyses en dehors des corrélations, le problème est d'un autre ordre. En effet, il

n'existe pas de théorie purement analytique pour le calcul des pertes dans une turbine et ces dernières sont calculées à partir de corrélations empiriques basées sur la géométrie et les caractéristiques de l'écoulement. Or, il arrive que certaines combinaisons de géométries et de conditions d'opérations soient en dehors des corrélations et ceci cause une mauvaise évaluation des pertes pouvant entraîner un échec de l'analyse par P1242. Ce type de problème est peu fréquent lors d'un design préliminaire car les ingénieurs définissent toujours des géométries bien adaptées au problème. Cependant, les géométries et les conditions de l'écoulement définies par un optimiseur ne font pas nécessairement du sens. Il faut donc toujours garder en tête que l'analyse peut échouer et qu'il faut gérer cette situation dans un contexte de design automatisé.

Tous les fichiers d'entrée et de sortie de P1242 respectent un format fixe. Il est donc possible de faire l'intégration pour un cas général dans le cadre du programme P1242.

2.1.3 Description de RotorDesigner

RotorDesigner est, comme son nom l'indique, le programme d'analyse des structures rotatives. C'est un programme 1-D et demi. Il fournit principalement la durée de vie d'un étage de turbine, le design des attaches et des ailettes, celui du disque et de l'épaulement et la marge de résistance à la survitesse. RD est basé sur la théorie des petites déformations. Il s'exécute en environ 0.5 seconde pour le cas étudié. Il est plutôt instable et il faut donc bien contraindre les entrées dans une région réalisable sans quoi le calcul échoue. De plus, RD ne s'exécute pas convenablement de n'importe quel poste de travail et il a donc fallu écrire des scripts pour l'exécuter au bon endroit sur le réseau. Il ne s'exécute donc pas nécessairement localement et comme le réseau n'est pas toujours stable et RD non plus, on est alors dans un environnement d'échecs potentiels (« crash environment »).

Dans certains cas, les déformations du disque sont dans le domaine plastique et alors RD donne des résultats erronés pour la marge de résistance à la survitesse. Lorsque ceci ce produit, un autre programme nommé P0831 est exécuté. Dans l'ensemble du projet, P0831 sera traité comme une partie de RD car l'un ne va pas sans l'autre.

2.1.4 Description de P0831

P0831 est un programme d'éléments finis 2-D. Il permet de calculer les déformations élastiques et plastiques dans un disque de turbine afin d'en déterminer la limite de résistance à la survitesse. Ce programme prend environ deux minutes à s'exécuter, ce qui est lent par rapport à tous les autres programmes utilisés dans le cadre de cette étude.

P0831 est stable du moment que la géométrie en entrée permet d'obtenir un maillage géométriquement et topologiquement conforme. On entend par là que les frontières de la géométrie sont fermées et ne s'intersectent pas. Ces deux conditions sont remplies explicitement grâce à la paramétrisation du disque et donc, P0831 est considéré stable pour ce projet.

2.1.5 Nomenclature d'un moteur

Tout au long de ce mémoire, il sera question d'un moteur d'hélicoptère. On présente ici la nomenclature attachée à ce type de moteur. On présente ensuite la nomenclature propre à la partie turbine.

Moteur complet

Chaque moteur est divisé en plusieurs parties par des plans virtuels numérotés (appelés aussi stations). Pour un moteur d'hélicoptère, la numérotation typique est illustrée à la figure 2.1. La définition de chaque station est fournie au tableau 2.1.

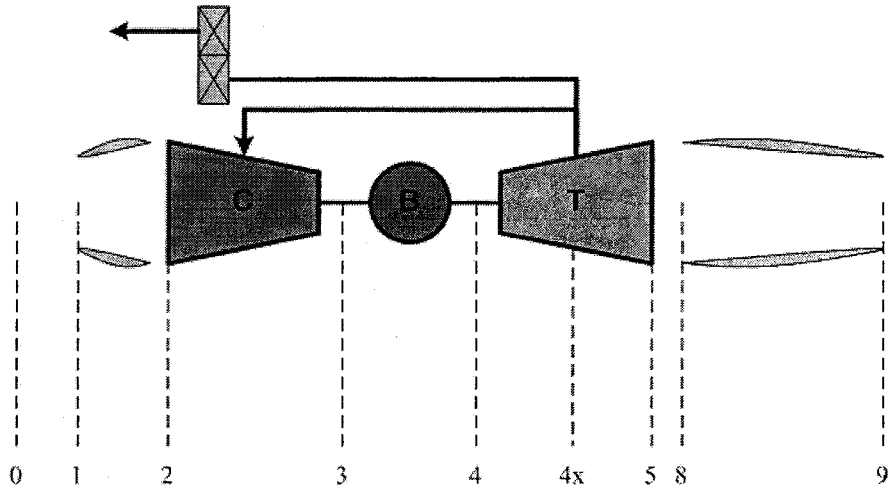

Figure 2.1 Nomenclature typique d'un moteur d'hélicoptère

Tableau 2.1 Description de la nomenclature d'un moteur d'hélicoptère

Plan	Description
0	écoulement libre
1	Entrée du moteur
2	Entrée du compresseur
3	Sortie du compresseur/entrée du brûleur
4	Sortie du brûleur/entrée de la turbine
4x	Sortie de la turbine du compresseur
5	Sortie de la turbine
8	Entrée de la tuyère d'échappement
9	Sortie de la tuyère d'échappement

Ce type de moteur comporte plusieurs parties dont les principales sont résumées au tableau 2.2.

Partie turbine

Puisque l'on étudie principalement la partie turbine dans ce projet, on présente ici la nomenclature propre à cette dernière. La figure 2.2 présente la numérotation typique de chaque station. Leur définition est fournie au tableau 2.3. Il est à noter

Tableau 2.2 Description des parties d'un moteur d'hélicoptère

Partie	Fonction	Stations
diffuseur	Ralentir l'écoulement et le pré comprimer	1-2
compresseur	Compresser l'écoulement pour améliorer la combustion	2-3
brûleur	Injecter de l'énergie dans le système	3-4
HPT	Extraire la puissance	4-4x
PT	Fournir la puissance au rotor de l'hélicoptère	4x-5
tuyère	Obtenir le maximum de poussée résiduelle	8-9

que cette numérotation est propre à chaque étage de la turbine.

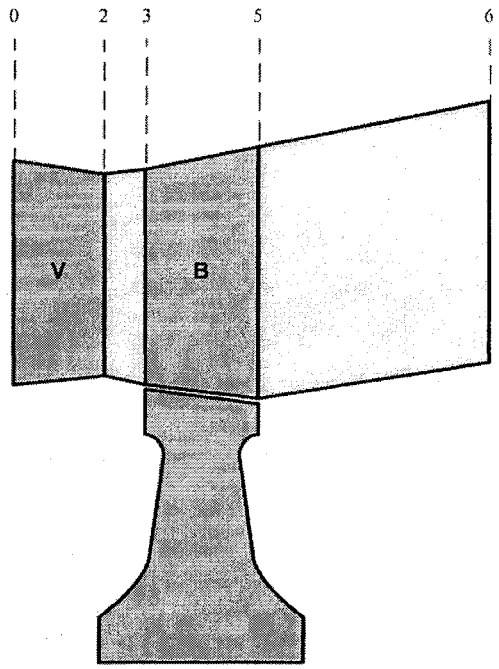

Figure 2.2 Nomenclature typique d'un étage de turbine

Chaque étage de turbine comporte 4 parties principales qui sont résumées au tableau 2.4

2.1.6 Définition du moteur étudié

Comme il a été mentionné à la sous-section 2.1.1, un seul moteur sera étudié dans ce projet. Un modèle SOAPP simplifié d'un « turboshaft » a été généré par M. John

Tableau 2.3 Description de la nomenclature d'une turbine

Plan	Description
0	Entrée de l'étage de turbine/entrée des aubes
2	Sortie des aubes
3	Entrée dans les ailettes
5	Sortie des ailettes
6	Sortie de l'étage

Tableau 2.4 Description des parties d'une turbine

Partie	Fonction	Stations
aube	Orienter l'écoulement	0-2
espace libre aubes ailettes	Séparer les aubes des ailettes	2-3
ailette	Faire tourner le rotor	3-5
conduit inter turbine	Connecter 2 turbines	5-6

Wheeler, ingénieur de P&WC. Ce modèle de moteur a été choisi pour deux raisons principales :

1. Le modèle SOAPP est simple et donc facile à comprendre et à manipuler.
2. Comme il s'agit d'une simplification d'un moteur, plusieurs données manquent. Le moteur étudié est donc fictif, car les données manquantes sont approximées à partir de l'expérience des ingénieurs. L'avantage de ce fait est qu'un plus grand nombre d'informations pourront être divulgué dans le présent mémoire.

Les données manquantes sont principalement géométriques. Ces données sont requises par P1242 et RD et sont donc approximées en modifiant des fichiers d'entrées existants.

La turbine du moteur ainsi créée est illustrée à la figure 2.3. Certaines caractéristiques du moteur sont données au tableau 2.5. Il est à noter que même le fichier SOAPP a été modifié afin de partir d'un point réalisable pour tous les programmes lors de l'optimisation. Comme on le voit, c'est un moteur avec deux étages de turbine. Le premier entraine le compresseur et le second entraine le rotor de l'hélicoptère.

Tableau 2.5 Quelques caractéristiques du moteur étudié

Variable	Description	Valeur	Unité
sfc	Consommation spécifique de carburant	7.588×10^{-5}	kg/kW.s
HPT η	Rendement de la turbine haute pression	0.8357	adim.
PT η	Rendement de la turbine de puissance	0.8673	adim.
$Power_{shaft}$	Puissance sur l'arbre de transmission	958	kW
W_{INLET}	Débit massique d'air à l'entrée du moteur	3.29	kg/s
PR	Ratio de pression du compresseur	12.9	adim.
HPT RPM_C	RPM corrigé de la turbine haute pression*	40000	tr/min
PT RPM	RPM de la turbine de puissance	28750	tr/min
HPT Δh	Variation d'enthalpie de la turbine haute pression	382.8	kJ/kg
PT Δh	Variation d'enthalpie de la turbine de puissance	290.2	kJ/kg
Poids	Poids total de tous les rotors	16.36	kg

* $RPM_C = \frac{RPM}{\sqrt{T_2/T_{ref}}}$

2.2 Définition du modèle mathématique complet

À ce stade-ci, la définition physique et la paramétrisation (voir l'annexe I) du problème sont complétées. Tous les éléments sont donc réunis pour débuter la définition du modèle mathématique du problème.

Voici quelques conventions nécessaires à la compréhension de cette section et des suivantes :

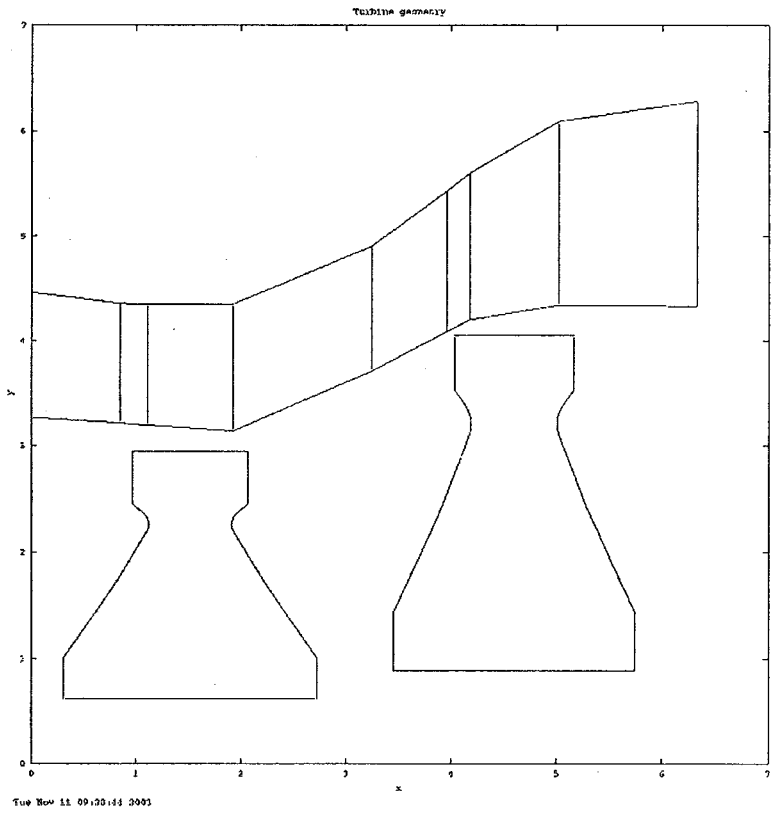

Figure 2.3 Géométrie du moteur étudié

N = nombre de discipline

D_i = programme d'analyse de la discipline i

F_i = fonction coût associée à la discipline i

C_i = vecteur des contraintes associées à la discipline i

s = vecteur des variables communes à deux disciplines ou plus (shared)

l_i = vecteur des variables propres à la discipline i (local)

p_i = vecteur des paramètres de la discipline i (parameter)

r_i = vecteur des résultats de la discipline i (result)

t_i = vecteur des variables de relaxation du couplage de la discipline i (tentative)

On divise les variables en catégories car les formulations FIO et DAO étudiées dans ce projet sont basées sur ces catégories.

Puisqu'il y a uniquement trois disciplines pour le problème étudié, alors $N = 3$ et $i = \{\text{SOAPP, P1242, RD}\}$. Il reste à définir les vecteurs s, l_i, t_i (ou p_i ou r_j), C_i et F_i. La dimension de ces vecteurs dépend des valeurs suivantes :

- nombre de turbines (m),
- nombre d'étages par turbine ($\vec{n}$),
- nombre de lobes par attache pour chaque étage ($\vec{o}$),
- nombre d'aires à définir pour chaque ailette ($\vec{p}$),
- nombre de températures à définir pour chaque ailette ($\vec{q}$).

Pour le cas présent, le nombre de turbines est de 2 ($m = 2$), il y a 1 étage par turbine ($\vec{n} = \{1 \quad 1\}$), 3 lobes par attache pour chaque étage ($\vec{o} = \{3 \quad 3\}$), 4 aires et 5 températures à définir pour chaque ailettes ($\vec{p} = \{4 \quad 4\}$) ($\vec{q} = \{5 \quad 5\}$). Le tableau 2.6 montre l'ensemble des variables, contraintes et objectifs pour le moteur étudié. L'annexe I fournit une description détaillée des variables présentées dans ce tableau. Pour comprendre la notation employée dans le tableau 2.6, il faut connaître la signification des indices suivants :

1. $[m]$ signifie, pour chaque turbine,
2. $[m, \vec{n}]$ signifie, pour chaque étage de chaque turbine,
3. $[m, \vec{n}, \vec{o}]$ signifie, pour chaque lobe des attaches de chaque étage de chaque turbine,
4. $[m, \vec{n}, \vec{p}]$ signifie, pour chaque aires des sections des ailettes de chaque étage de chaque turbine,
5. $[m, \vec{n}, \vec{q}]$ signifie, pour chaque température des sections des ailettes de chaque étage de chaque turbine.

Tableau 2.6 Description des ensembles de variables, de contraintes et d'objectifs

Ensemble	Variable	Description
$s =$	$RPM_{[2]}$	RPM de la turbine 2 (PT)
	$1^{st}Stage_{AN^2}$	conduite de gaz (voir l'équation I.4)
	$1^{st}Stage_{RimSpeed}$	conduite de gaz (voir l'équation I.4)
	$B_{x-v_{[m,\vec{n}]}}$	corde axiale des aubes (voir la figure I.1)
	$B_{x-b_{[m,\vec{n}]}}$	corde axiale des ailettes (voir la figure I.1)
	$\theta_{b-h_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{b-t_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{v-h_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{v-t_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{AL56-h_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{AL56-t_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$Gap_{h_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$Gap_{t_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$AL_{56_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$NOB_{[m,\vec{n}]}$	nombre d'ailettes
$l_{SOAPP} =$	W_C	débit corrigé en entrée du moteur
	PR	ratio de pression du compresseur
	RPM_C	RPM corrigé de la turbine 1 (HPT)
$l_{P1242} =$	$\Delta Reaction_{[m,\vec{n}]}$	voir l'équation 4.1
	$NOV_{[m,\vec{n}]}$	nombre d'aubes
$l_{RD} =$	$BladeMaterial_{[m,\vec{n}]}$	matériau des ailettes
	$DiscMaterial_{[m,\vec{n}]}$	matériau du disque
	$Shrouded_{[m,\vec{n}]}$	épaulement (oui ou non)
	$Platform_{[m,\vec{n}]}$	designer la plateforme (oui ou non)
	$redlinerpm_{[m,\vec{n}]}$	RPM de design pour RD
	$overhanglength_{[m,\vec{n}]}$	plateforme (voir la figure I.3)
	$upwallthickness_{[m,\vec{n}]}$	plateforme (voir la figure I.3)
	$pedestalthickness_{[m,\vec{n}]}$	plateforme (voir la figure I.3)
	$platformthickness_{[m,\vec{n}]}$	plateforme (voir la figure I.3)

continue à la page suivante

suite de la page précédente

Ensemble	Variable	Description
	$matdepreciation_{[m,\vec{n}]}$	facteur de dégradation du matériel de l'ailette
	$shroudedbendstress_{[m,\vec{n}]}$	contrainte en flexion dans l'épaulement
	$tipaxialchord_{[m,\vec{n}]}$	ailettes (voir la figure I.2)
	$shroudscalefactor_{[m,\vec{n}]}$	facteur d'échelle de l'épaulement
	$areaatrad_{[m,\vec{n},\vec{p}]}$	aire de l'ailette (voir l'équation I.5)
	$topsection_{[m,\vec{n}]}$	cavité dans l'ailette (voir le tableau I.2)
	$topcavityarea_{[m,\vec{n}]}$	cavité dans l'ailette (voir le tableau I.2)
	$midsection_{[m,\vec{n}]}$	cavité dans l'ailette (voir le tableau I.2)
	$midcavityarea_{[m,\vec{n}]}$	cavité dans l'ailette (voir le tableau I.2)
	$endsection_{[m,\vec{n}]}$	cavité dans l'ailette (voir le tableau I.2)
	$endcavityarea_{[m,\vec{n}]}$	cavité dans l'ailette (voir le tableau I.2)
	$caarea_{[m,\vec{n}]}$	aire de passage de l'air de refroidissement dans les attaches
	$wedgeangle_{[m,\vec{n}]}$	attaches (voir la figure I.4)
	$drloadangle_{[m,\vec{n}]}$	attaches (voir la figure I.4)
	$drunloadangle_{[m,\vec{n}]}$	attaches (voir la figure I.4)
	$topbladeangle_{[m,\vec{n}]}$	attaches (voir la figure I.4)
	$topneckwidth_{[m,\vec{n}]}$	attaches (voir la figure I.4)
	$rlobe_{[m,\vec{n},\vec{o}]}$	attaches (voir la figure I.4)
	$rneck_{[m,\vec{n},\vec{o}]}$	attaches (voir la figure I.4)
	$offset_{[m,\vec{n},\vec{o}]}$	attaches (voir la figure I.4)
	$rimwidth_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$broachangle_{[m,\vec{n}]}$	plateforme (voir la figure I.3)
	$fixing_{r2_{[m,\vec{n}]}}$	
	$bottomlobeangle_{[m,\vec{n}]}$	attaches (voir la figure I.4)
	$bottomloberadius_{[m,\vec{n}]}$	attaches (voir la figure I.4)
	$thickness_{[m,\vec{n}]}$	disque (voir la figure I.5, symbole T)
	$rivetradius_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$rivet_{a1_{[m,\vec{n}]}}$	
	$highneckheight_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$highneckwidth_{[m,\vec{n}]}$	disque (voir la figure I.5)

continue à la page suivante

suite de la page précédente

Ensemble	Variable	Description
	$lowneckheight_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$lowneckwidth_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$lowneckrad_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$highneckrad_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$discboreradius_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$discborewidth_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$discboreheight_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$MUF_{[m,\vec{n}]}$	Facteur d'utilisation du matériel du disque
$t_{SOAPP} =$	W_0	débit de gaz en entrée de la turbine
	T_4	température à l'entrée de la turbine
	p_4	pression à l'entrée de la turbine
	fa	ratio carburant-air
	$\Delta h_{[m]}$	variation d'enthalpie de la turbine
	BTT_3	Température à la sortie du compresseur en condition de « Block Test »
	BTT_4	Température à l'entrée de la turbine en condition de « Block Test »
	$RPM_{[1]}$	RPM de la première turbine (HPT)
	BTW	débit de gaz à l'entrée de la turbine en condition de « Block Test »
$t_{P1242} =$	$Swirl_{MN}$	nombre de Mach à la sortie des ailettes du dernier étage
	$Swirl_{Angle}$	angle de l'écoulement à la sortie des ailettes du dernier étage
	$\eta_{[m]}$	rendement des turbines
	$\Delta p/p_{[m,\vec{n}]}$	perte de pression des turbines
	$BLWA_7$	débit d'air de refroidissement injecté au bord d'attaque des ailettes
	$BLWA_9$	débit d'air de refroidissement injecté au bord de fuite des ailettes
	$Tref_{design_{[m,\vec{n}]}}$	température de référence pour le design

continue à la page suivante

suite de la page précédente

Ensemble	Variable	Description
		des ailettes
	$Trel_{design_{[m,\vec{n}]}}$	température relative pour le design des ailettes
	$Tref/Trel_{[m,\vec{n},\vec{q}]}$	ratio de $Tref/Trel$ à différentes sections radiales de l'ailette
$t_{RD} =$	$\emptyset$	
$C_{SOAPP} =$	$\emptyset$	
$C_{P1242} =$	$ZwV_{[m,\vec{n}]}$	charge aérodynamique sur les aubes
	$ZwB_{[m,\vec{n}]}$	charge aérodynamique sur les ailettes
	α_{55}	angle absolu de l'écoulement à la sortie des ailettes
	Mn_{55}	nombre de Mach absolu à la sortie des ailettes
	$\alpha_{[m,\vec{n},6]}$	angle relatif de l'écoulement à la sortie de chaque station de la turbine
	$PS_{in}PS_{out_{[m,\vec{n},2]}}$	ratio de pression de chaque turbine
	$ECDA_{[m,\vec{n}]}$	facteur géométrique de la conduite de gaz
$C_{RD} =$	$bladenominalstress_{[m,\vec{n},\vec{o}]}$	contrainte nominale dans les attaches
	$bladepeakstress_{[m,\vec{n},\vec{o}]}$	contrainte maximale dans les attaches
	$bladebearingstress_{[m,\vec{n},\vec{o}]}$	contrainte de contact dans les attaches
	$bladeshearstress_{[m,\vec{n},\vec{o}]}$	contrainte de cisaillement dans les attaches
	$bladebendingstress_{[m,\vec{n},\vec{o}]}$	contrainte de flexion dans les attaches
	$discnominalstress_{[m,\vec{n},\vec{o}]}$	contrainte nominale dans les disques
	$discpeakstress_{[m,\vec{n},\vec{o}]}$	contrainte maximale dans les disques
	$discbearingstress_{[m,\vec{n},\vec{o}]}$	contrainte de contact dans les disques
	$discshearstress_{[m,\vec{n},\vec{o}]}$	contrainte de cisaillement dans les disques
	$discbendingstress_{[m,\vec{n},\vec{o}]}$	contrainte de flexion dans les disques
	$discburstmargin_{[m,\vec{n},\vec{o}]}$	marge de résistance à la survitesse des disques
	$biaxiallimit_{[m,\vec{n}]}$	limite de biaxialité des contraintes dans le disque
	$discborestress_{[m,\vec{n}]}$	contraintes maximale au rayon interne du disque
	$BTAirfoilCreepLife_{[m,\vec{n}]}$	vie en fatigue à basse fréquence (LCF) des ailettes en condition de « block test »
	$BTShroudCreepLife_{[m,\vec{n}]}$	vie en fatigue à basse fréquence (LCF) des

continue à la page suivante

suite de la page précédente

Ensemble	Variable	Description
		épaulement en condition de « block test »
	$MixedAirfoilCreepLife_{[m,\vec{n}]}$	vie en fatigue à basse fréquence (LCF) des ailettes
	$MixedShroudCreepLife_{[m,\vec{n}]}$	vie en fatigue à basse fréquence (LCF) des épaulements
$F_{SOAPP} =$	sfc	consommation spécifique de carburant (voir l'équation 3.2)
$F_{P1242} =$	$\eta_{[m]}$	rendement des turbines
$F_{RD} =$	$totalweight_{[m,\vec{n}]}$	poids total des rotors

2.3 Mise à l'échelle

Pour que l'optimisation avec des méthodes à gradients fonctionne bien, il faut que toutes les valeurs soient mises à l'échelle. Selon Papalambros [29], « la mise à l'échelle est le facteur le plus important, mais le plus simple, pouvant faire la différence entre le succès et l'échec d'une procédure d'optimisation ». En effet, si tel n'est pas le cas, il est très difficile de trouver un pas de différenciation qui convient pour toutes les variables. De plus, une différence de plusieurs ordre de grandeur entre les variables peut mener à une matrice hessienne mal conditionnée, donc à des problèmes numériques.

Pour le cas présent, la valeur des variables varie entre environ 0.005 pour les pertes de charge $\Delta p/p_{[m,\vec{n}]}$ et 5×10^{10} pour le $1^{st}Stage_{AN^2}$. Il est donc essentiel d'adimensionnaliser le problème avant de conduire des optimisations avec des méthodes à gradients.

2.3.1 Choix du type de mise à l'échelle

Il existe plusieurs façon de mettre à l'échelle un problème. La plus simple est de diviser chaque valeur par une valeur de référence, typiquement la valeur initiale non signée. Cette approche est correcte si les variables ne varient pas trop par rapport à

leur valeur initiale car le but d'une mise à l'échelle est d'obtenir des valeurs adimensionnelles qui sont toutes du même ordre de grandeur.

Dans ce projet, on part d'une solution initiale réalisable et on fait l'hypothèse qu'elle est relativement bonne. En conséquence, les variables ne varieront pas beaucoup lors du processus d'optimisation. En fait, toutes les variables à l'exception des variables couplées seront contraintes dans un intervalle de $\pm 15\%$ (voir le chapitre 3). Les variables couplées ne varient toutefois pas beaucoup, elles non plus. On opte donc pour la mise à l'échelle simple énoncée ci-dessus.

2.3.2 Mise à l'échelle des variables

La mise à l'échelle des variables va donc être réalisée en divisant simplement toutes les variables par leur valeur initiale.

$$\widetilde{x} = \frac{x}{|x_{init}|} \tag{2.1}$$

Si $x_{init} = 0$, on adimentionalise pas la variable donc son facteur d'échelle est 1, tel que celui par défaut dans iSIGHT. Les variables mises à l'échelles auront donc une valeur absolue proche de l'unité.

Cette mise à l'échelle a d'abord été utilisée pour toutes les variables. Il a alors été observé que la fonction coût et les contraintes étaient pratiquement insensibles à certaines variables pour le pas de différenciation choisi. Or, ce sont ces variables qui déterminent la forme de la conduite de gaz et elles devraient avoir une influence. Afin de corriger ce problème, le facteur d'échelle de ces variables a été modifié afin que leur dérivée mise à l'échelle soit plus grande. Le raisonnement est donné à l'équation

2.2, où SF_x indique le facteur de mise à l'échelle pour la variable x.

$$\begin{aligned} \widetilde{f} &= \frac{f}{SF_f} \\ \widetilde{x} &= \frac{x}{SF_x} \end{aligned} \quad \Rightarrow \quad \frac{\partial \widetilde{f}}{\partial \widetilde{x}} = \frac{\partial \left(\frac{f}{SF_f}\right)}{\partial \left(\frac{x}{SF_x}\right)} = \frac{SF_x}{SF_f}\frac{\partial f}{\partial x} \tag{2.2}$$

On voit donc que pour augmenter $\frac{\partial \widetilde{f}}{\partial \widetilde{x}}$, il faut augmenter SF_x ou diminuer SF_f. On choisit d'augmenter SF_x car cela n'influence que les dérivées de la variable x tandis que si on diminue SF_f on influence toutes les dérivées de f. Les variables x qui sont ainsi mises à l'échelle sont présentées au tableau 2.7 avec leur facteur d'échelle SF_x.

Tableau 2.7 Variables dont le facteur d'échelle a été ajusté

Variables x	Facteur d'échelle SF_x	Variables x	Facteur d'échelle SF_x
θ_{v-h_i}	100	Gap_{h_i}	0.5
θ_{v-t_i}	100	B_{xv_i}	2
θ_{b-h_i}	100	B_{xb_i}	2
θ_{b-t_i}	100	AL_{5-6_i}	3
θ_{AL56t_i}	100	$rimwidth_i$	10
θ_{AL56h_i}	100	$broachangle_i$	35
Gap_{t_i}	0.5		

2.3.3 Mise à l'échelle des contraintes

Pour les contraintes de compatibilité $(t - r = 0)$ relatives à la stratégie d'optimisation DAO (voir la sous-section 3.3.2), les valeurs initiales sont nulles car on débute l'optimisation à partir d'un point réalisable. On les divisera donc par les t initiaux non signés.

$$\widetilde{t - r} = \frac{t - r}{|t_{init}|} \tag{2.3}$$

L'avantage de cette méthode est que l'on ramène toutes les contraintes au même ordre de grandeur. En effet, il faut se rappeler que les variables ont des ordres de grandeur très différents et donc que les différences $t_i - r_i$ auront aussi des ordres de grandeur différents. En prenant comme valeurs mises à l'échelle les différences relatives, on élimine ainsi ce problème.

Pour ce qui est des autres contraintes, on les divises simplement par leur valeur initiale non signée :

$$\widetilde{C} = \frac{C}{|C_{init}|} \tag{2.4}$$

2.3.4 Mise à l'échelle des objectifs

On effectue également la mise à l'échelle des objectifs. On obtiendra ainsi le même ordre de grandeur pour l'ensemble des variables, des contraintes et des objectifs. Il sera alors plus facile de donner les bons poids à chacun afin d'atteindre de façon efficace le minimum. Chaque objectif F_i est divisé par sa valeur initiale non signée :

$$\widetilde{F} = \frac{F}{|F_{init}|} \tag{2.5}$$

Les objectifs mis à l'échelle ont donc une valeur absolue proche de l'unité.

2.4 Réduction du problème

Le problème défini jusqu'à présent comporte un grand nombre de variables et de contraintes. Certaines hypothèses ont été apportées afin de le simplifier. De plus, une fois le problème complètement intégré dans iSIGHT, il a été possible de faire des tests en utilisant les fonctionnalités d'iSIGHT afin de réduire d'avantage le nombre de variables.

2.4.1 Hypothèses simplificatrices

Étant donné le nombre élevé de variables, des moyens de réduire ce nombre ont été explorés pour faciliter la résolution du problème d'optimisation.

- Les variables entières suivantes : NOV_i et NOB_i, qui représentent respectivement le nombre d'aubes et d'ailettes ont été laissé de côté. En éliminant les variables entières, on simplifie grandement le problème. En effet, il est difficile de traiter des problèmes contenant des variables entières avec une méthode à gradients. On se facilite ainsi beaucoup la tâche mais le prix à payer est qu'on ne peut pas varier le nombre d'ailettes et que ce nombre a une influence non négligeable pour l'aérodynamique et les structures rotatives. Certains essais ont été tentés où le nombre d'ailettes était traité comme une variable réelle mais sans succès. Le nombre d'ailettes a donc été fixé selon l'expérience des ingénieurs et restera fixe dans ce projet.
- On suppose qu'il n'y a pas de cavité dans les ailettes. Les variables suivantes sont alors éliminées : $[top|mid|end]section_i$ et $[top|mid|end]cavityarea_i$. Ceci permet d'éliminer toutes les autres variables entières ($[top|mid|end]section_i$).
- Le premier étage de turbine est refroidi et n'a pas d'épaulement. Pour le deuxième étage, c'est le contraire. Ceci est normal car les étages refroidis n'ont presque jamais d'épaulement et il n'y a pas de raison, à priori, de ne pas en mettre pour les non-refroidis.
- On assume également que le rendement du compresseur est constant même si on fait varier les variables de type l_{SOAPP} : PR, W_C et RPM_C. Ceci est inexact mais il n'y a pas actuellement de programme d'analyse robuste de disponible pour le compresseur. Afin de diminuer les effets de cette limitation, on demande à SOAPP d'utiliser le rendement polytropique car ce dernier est moins sensible au rapport de pression du compresseur (PR) que le rendement isentropique (voir la figure 2.4). Par contre, on sait que cette variable a beaucoup d'influence sur le *sfc* et ceci constitue une limitation de la présente intégration. Pour ce

qui est de RPM_C et de W_C, tel que discuté précédemment, on ne permet pas à l'optimiseur de les faire varier de plus de $\pm 15\%$. En général, ces variables sont limitées respectivement par les contraintes structurelles et aérodynamiques.

- Afin de ne pas être restreint quant au choix de la méthode d'optimisation, on élimine également toutes les variables discrètes : $BladeMaterial$, $DiscMaterial$, $Shrouded$, $Platform$.

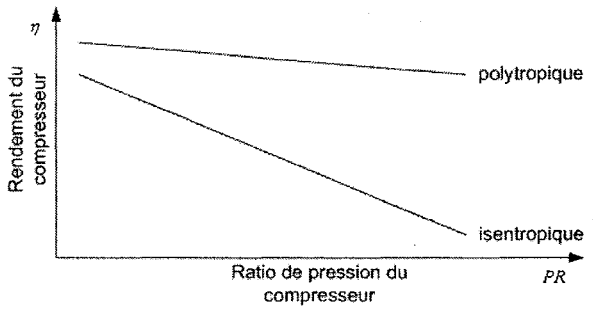

Figure 2.4 Sensibilité des rendements isentropique et polytropique du compresseur envers le ratio de pression (PR)

On obtient finalement un problème qui contient uniquement des variables réelles. Le fait d'éliminer ces variables a peu d'influence sur le design car il est assez facile pour les ingénieurs du département de structures rotatives de déterminer quels matériaux sont appropriés pour une turbine donnée. Le fait de mettre ou de ne pas mettre un épaulement dépend de règles empiriques et on demande toujours à RD de faire le design de la plateforme.

Finalement, on impose à RD de ne pas exécuter P0831 lors du calcul des gradients par l'optimiseur. La raison est que P0831 met environ 2 minutes à s'exécuter et ceci ralentit beaucoup trop l'optimisation étant donné le grand nombre d'exécutions de RD. On ne compromet toutefois pas beaucoup le résultat de l'optimisation car P0831 ne fait que fournir une valeur plus précise de certaines sorties.

2.4.2 Observation du comportement de NLPQL

Afin d'arriver à obtenir une solution de qualité en un temps raisonnable, plusieurs ajustements ont dû être faits lors des optimisations préliminaires. Ceci a permis d'observer que certaines variables ne sont pratiquement pas modifiées par l'optimiseur. Il n'était pas normal que certaines de ces variables ne varient pas et des mesures ont été prises en ce sens par la mise à l'échelle (voir la section 2.3.2). Par contre, certaines variables semblent réellement avoir peu d'influence sur le résultat de l'optimisation, peu importe les situations. Ces variables ont été identifiées et retirées de l'optimisation.

Toutes les variables retirées sont relatives aux structures rotatives et notamment aux attaches des ailettes. En effet, il y a beaucoup de variables pour les attaches mais l'optimiseur ne les varies pas, probablement parce que leur gradient est trop faible par rapport aux autres. Afin de faire varier ces variables, une sous-optimisation concernant seulement les attaches pourrait être envisagé. Cependant, ceci n'a pas été jugé nécessaire dans ce projet. Par conséquent, l'optimisation sera limitée par la capacité des attaches initiales à reprendre la charge. L'optimum trouvé ne sera donc pas absolu puisque la solution finale sature habituellement les contraintes dans les attaches et que ces dernières ne sont pas optimales puisque l'optimiseur ne fait pas varier leur géométrie. Ceci constitue une seconde limitation du présent travail.

2.4.3 Définition du modèle mathématique simplifié

Suite aux simplifications apportées et aux informations présentées aux sections 2.2 et 2.4.2, il est maintenant possible de tracer le modèle mathématique simplifié de la turbine. On présente donc ici l'ensemble des variables, des contraintes et l'objectif mis en jeu dans ce projet.

Tableau 2.8 Description des ensembles de variables, de contraintes et d'objectifs

Ensemble	Variable	Description
$s =$	$RPM_{[2]}$	RPM de la turbine 2 (PT)
	$1^{st}Stage_{AN^2}$	conduite de gaz (voir l'équation I.4)
	$1^{st}Stage_{RimSpeed}$	conduite de gaz (voir l'équation I.4)
	$B_{x-v_{[m,\vec{n}]}}$	corde axiale des aubes (voir la figure I.1)
	$B_{x-b_{[m,\vec{n}]}}$	corde axiale des ailettes (voir la figure I.1)
	$\theta_{b-h_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{b-t_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{v-h_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{v-t_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{AL56-h_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$\theta_{AL56-t_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$Gap_{h_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$Gap_{t_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
	$AL_{56_{[m,\vec{n}]}}$	conduite de gaz (voir la figure I.1)
$l_{SOAPP} =$	W_C	débit corrigé en entrée du moteur
	PR	ratio de pression du compresseur
	RPM_C	RPM corrigé de la turbine 1 (HPT)
$l_{P1242} =$	$\Delta Reaction_{[m,\vec{n}]}$	voir l'équation 4.1
$l_{RD} =$	$rimwidth_{[m,\vec{n}]}$	disque (voir la figure I.5)
	$broachangle_{[m,\vec{n}]}$	plateforme (voir la figure I.3)
$t_{SOAPP} =$	W_0	débit de gaz en entrée de la turbine
	T_4	température à l'entrée de la turbine
	p_4	pression à l'entrée de la turbine
	fa	ratio carburant-air
	$\Delta h_{[m]}$	variation d'enthalpie de la turbine
	BTT_3	Température à la sortie du compresseur en condition de « Block Test »
	BTT_4	Température à l'entrée de la turbine en condition de « Block Test »

continue à la page suivante

suite de la page précédente

Ensemble	Variable	Description
	$RPM_{[1]}$	RPM de la première turbine (HPT)
	BTW	débit de gaz à l'entrée de la turbine en condition de « Block Test »
$t_{P1242} =$	$Swirl_{MN}$	nombre de Mach à la sortie des ailettes du dernier étage
	$Swirl_{Angle}$	angle de l'écoulement à la sortie des ailettes du dernier étage
	$\eta_{[m]}$	rendement des turbines
	$\Delta p/p_{[m,\vec{n}]}$	perte de pression des turbines
	$BLWA_7$	débit d'air de refroidissement injecté au bord d'attaque des ailettes
	$BLWA_9$	débit d'air de refroidissement injecté au bord de fuite des ailettes
	$Tref_{design_{[m,\vec{n}]}}$	température de référence pour le design des ailettes
	$Trel_{design_{[m,\vec{n}]}}$	température relative pour le design des ailettes
	$Tref/Trel_{[m,\vec{n},\vec{q}]}$	ratio de $Tref/Trel$ à différentes sections radiales de l'ailette
$t_{RD} =$	$\emptyset$	
$C_{SOAPP} =$	$\emptyset$	
$C_{P1242} =$	$ZwV_{[m,\vec{n}]}$	charge aérodynamique sur les aubes
	$ZwB_{[m,\vec{n}]}$	charge aérodynamique sur les ailettes
	α_{55}	angle absolu de l'écoulement à la sortie des ailettes
	Mn_{55}	nombre de Mach absolue à la sortie des ailettes
	$\alpha_{[m,\vec{n},6]}$	angle relatif de l'écoulement à la sortie de chaque station de la turbine
	$PS_{in}PS_{out_{[m,\vec{n},2]}}$	ratio de pression de chaque turbine
	$ECDA_{[m,\vec{n}]}$	facteur géométrique de la conduite de gaz
$C_{RD} =$	$bladenominalstress_{[m,\vec{n},\vec{o}]}$	contrainte nominale dans les attaches
	$bladepeakstress_{[m,\vec{n},\vec{o}]}$	contrainte maximale dans les attaches

continue à la page suivante

suite de la page précédente

Ensemble	Variable	Description
	$bladebearingstress_{[m,\vec{n},\vec{o}]}$	contrainte de contact dans les attaches
	$bladeshearstress_{[m,\vec{n},\vec{o}]}$	contrainte de cisaillement dans les attaches
	$bladebendingstress_{[m,\vec{n},\vec{o}]}$	contrainte de flexion dans les attaches
	$discnominalstress_{[m,\vec{n},\vec{o}]}$	contrainte nominale dans les disques
	$discpeakstress_{[m,\vec{n},\vec{o}]}$	contrainte maximale dans les disques
	$discbearingstress_{[m,\vec{n},\vec{o}]}$	contrainte de contact dans les disques
	$discshearstress_{[m,\vec{n},\vec{o}]}$	contrainte de cisaillement dans les disques
	$discbendingstress_{[m,\vec{n},\vec{o}]}$	contrainte de flexion dans les disques
	$discburstmargin_{[m,\vec{n},\vec{o}]}$	marge de résistance à la survitesse des disques
	$biaxiallimit_{[m,\vec{n}]}$	limit de biaxialité des contraintes dans les disques
	$discborestress_{[m,\vec{n}]}$	contraintes maximale au rayon interne du disque
	$BTAirfoilCreepLife_{[m,\vec{n}]}$	vie en fatigue à basse fréquence (LCF) des ailettes en condition de « block test »
	$BTShroudCreepLife_{[m,\vec{n}]}$	vie en fatigue à basse fréquence (LCF) des épaulement en condition de « block test »
	$MixedAirfoilCreepLife_{[m,\vec{n}]}$	vie en fatigue à basse fréquence (LCF) des ailettes
	$MixedShroudCreepLife_{[m,\vec{n}]}$	vie en fatigue à basse fréquence (LCF) des épaulement
$F_{SOAPP} =$	sfc	consommation spécifique de carburant (voir l'équation 3.2)
$F_{P1242} =$	$\emptyset$	
$F_{RD} =$	$\emptyset$	

On est maintenant en mesure d'évaluer la taille du problème pour FIO et pour DAO. Le tableau 2.9 présente une comparaison entre la taille avant et après réduction pour les deux stratégies.

Comme on le voit clairement à la lecture du tableau 2.9, le problème est beaucoup plus petit une fois réduit. Le nombre de variables chute de 77% pour FIO et de 64% pour DAO. On observe aucune réduction du nombre de contraintes et une diminution

de 80% du nombre d'objectifs dans les deux cas. Finalement, il n'y a plus de variables discrètes et entières et ceci va grandement faciliter l'optimisation.

Tableau 2.9 Taille du problème avant et après réduction

			complet			réduit		
			réelle	entière	discrète	réelle	entière	discrète
FIO	variables		130	10	8	34	0	0
	contraintes	$=$	0	0	0	0	0	0
		$\leq$	112	0	0	112	0	0
	objectifs		5	0	0	1	0	0
DAO	variables		170	10	8	66	0	0
	contraintes	$=$	32	0	0	32	0	0
		$\leq$	112	0	0	112	0	0
	objectifs		5	0	0	1	0	0

CHAPITRE 3

OPTIMISATION

Ce chapitre a pour but de modéliser le problème d'optimisation avec les stratégies FIO et DAO. On peut exprimer tout problème d'optimisation en nombres réels par la forme standard suivante :

$$\begin{aligned} &\min_{\vec{x}\in\mathbb{R}^n} \vec{F}(\vec{x}) \\ &\vec{G}(\vec{x}) \leq 0 \\ &\vec{H}(\vec{x}) = 0 \\ &\vec{x}_{min} \leq \vec{x} \leq \vec{x}_{max} \end{aligned} \tag{3.1}$$

Le chapitre précédent a présenté la modélisation mathématique de la turbine à gaz. Ceci a permis de définir $\vec{x}$, $\vec{G}(\vec{x})$, $\vec{H}(\vec{x})$ et $\vec{F}(\vec{x})$. En effet, $\vec{x}$ correspond aux variables de type s, l, t et le vecteur $\vec{G}$ correspond aux contraintes de type C du tableau 2.8. Au départ, le modèle du problème ne contient pas de contraintes d'égalités ($\vec{H}$) et $\vec{F}$ correspond à l'objectif (*sfc*). On verra toutefois que la méthode de décomposition DAO requiert l'introduction de contraintes d'égalités de type $t-r=0$, ce qui requiert l'utilisation de la contrainte $\vec{H}$ qui a été indirectement définie au chapitre précédent.

La première étape de ce chapitre sera donc de compléter le modèle mathématique de l'optimisation. On décrit dans un premier temps plus en détails l'objectif de l'optimisation $\vec{F}$. Par la suite, on présente une description des bornes $\vec{x}_{min}$ et $\vec{x}_{max}$ imposées aux variables, suivi des stratégies FIO et DAO, complétant ainsi l'écriture du problème d'optimisation. Puisque le but de tout ce travail est l'optimisation d'une turbine à gaz, les sections suivantes présenteront les optimiseurs employés pour résoudre les problèmes d'optimisation FIO et DAO.

3.1 Objectif d'optimisation

Plusieurs objectifs peuvent être choisis pour faire l'optimisation d'une turbine à gaz. La requête la plus fréquente des avionneurs est habituellement d'avoir un moteur avec une consommation spécifique de carburant (*sfc*) et un poids le plus petit possible, avec bien sûr un faible coût. Dans ce projet, on a choisi d'optimiser avec un seul objectif afin de concentrer les efforts sur la décomposition et ainsi s'affranchir des difficultés supplémentaires liées à l'optimisation multi-objectif. L'objectif choisi est le *sfc*, qui est un résultat du programme SOAPP. Le *sfc* représentent un des critères clés qui définissent les performances du moteur. De plus, il est possible pour les ingénieurs d'exprimer le coût, le poids, la durabilité et le rendement d'un moteur en terme de *sfc* équivalent. Pour le cas présent, la définition traditionelle du *sfc* sera utilisée sans ajout de calculs des *sfc* équivalents. On définit donc le *sfc* comme suit :

$$\vec{F} = sfc = \frac{\text{débit de carburant}}{\text{puissance nette}} \quad \left[\frac{kg}{kW.s}\right] \tag{3.2}$$

3.2 Imposition des bornes sur les variables

Par défaut, les bornes sur les variables ont été fixées à $\pm 15\%$ (voir la sous-section 2.3.2). Ceci n'est cependant pas le cas pour toutes les variables. Les variables traitées de façon spéciale sont listées au tableau 3.1. On doit en effet imposer des bornes différentes sur certaines variables afin d'obtenir une solution qui a un sens physique. Par exemple, on sait que l'optimiseur va toujours saturer le rapport de pression du compresseur (PR) à sa borne supérieure puisque ceci diminue le *sfc*. Ceci est relié au fait que le rendement du compresseur n'est pas ajusté en conséquence. Il faut donc limiter la variation du PR afin de rester dans une zone où le rendement polytropique du compresseur varie peu.

Par contre, pour d'autres variables comme le RPM_2 ou les dimensions axiales comme B_x et AL_{5-6_i}, il n'y a pas vraiment de raison de les borner explicitement

selon les critères de design établis pour ce projet. On impose donc des bornes plus larges pour ces variables afin de s'assurer qu'elle ne seront pas saturées à l'optimum.

Certains angles ne doivent pas excéder une certaine valeur pour des raisons aérodynamiques et structurelles connues des ingénieurs. En effet, sur le plan aérodynamique, P1242 à des faiblesses car il n'est qu'une approximation de la réalité et ceci peut mener l'optimiseur dans une région de l'espace de design qui ne fait pas de sens physique. On borne donc les θ_{x-x_i} dans une région physiquement possible.

On vient également limiter la borne supérieure du *rimwidth* car la valeur de cette variable ne doit pas dépasser un certain pourcentage de la largeur des ailettes (B_{x-b}). Finalement, on vient ajuster les bornes des variables de type t en se basant sur les résultats préliminaires de l'optimisation avec FIO. On s'assure ainsi que ces variables puissent varier suffisamment pour atteindre le même minimum que FIO.

Tableau 3.1 Variables dont les contraintes de borne ne sont pas de ±15%

Variables	Borne minimale (%)	Borne maximale (%)
PR	-15	8.5
RPM_2	-15	28
θ_{v-h_i}	{-2 -135}	{1287 30}
θ_{v-t_i}	{-113 -72}	{7 10}
θ_{b-h_i}	{-290 -25}	15
θ_{b-t_i}	{0 -30}	0
θ_{AL56-t_i}	{-12 -160}	{10 80}
θ_{AL56-h_i}	{-14 -3332}	{7 5782}
Gap_{t_i}	{-3 -1}	{15 67}

continue à la page suivante

suite de la page précédente

Variables	Borne minimale (%)	Borne maximale (%)
Gap_{h_i}	{-0.5 -0.5}	{500 200}
B_{x-v_i}	{-41 -31}	{53 81}
B_{x-b_i}	{-39 -41}	{59 53}
AL_{5-6_i}	{-62 -61}	{53 54}
rimwidth	{-25 -25}	1

3.3 Découplage du problème MDA

Dans le présent projet, il y a un couplage entre SOAPP et P1242. Il vient du fait que certaines entrées (p_i) sont constituées de composantes des vecteurs de sorties (r_j) d'une autre discipline (voir la figure 3.1). Les r_i s'obtiennent donc en résolvant le couplage suivant :

$$r_i = D_i\,(s,\ l_i,\ r_j) \quad i = 1..N,\ j = 1..N,\ j \neq i \tag{3.3}$$

où r_j correspond au vecteur p_i.

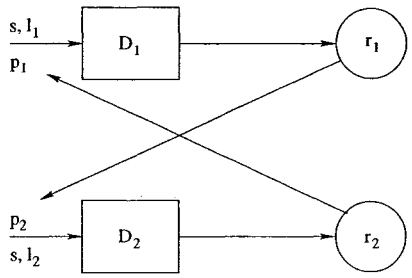

Figure 3.1 Couplage entre 2 disciplines

L'équation 3.3 définit ainsi un sytème à résoudre, nommé MDA (MultiDisciplinary Analysis) dans la littérature. Ce système MDA doit être résolu afin d'obtenir les bonnes valeurs de sortie. On dénote ces dernières par r_i^* dans le reste de ce travail

afin de distinguer les valeurs de sortie des programmes d'analyses (r_i) des valeurs de sorties convergées, soit celles obtenues par la résolution du système MDA (r_i^*). Il est important de noter que chaque r_i^* dépend explicitement de $s,\ l_1,\ l_2,\ \ldots,\ l_i,\ \ldots,\ l_N$ tandis que chaque r_i dépend de $s,\ l_i,\ r_1,\ r_2,\ \ldots,\ r_j,\ \ldots,\ r_N$ où $j \neq i$. Il y a plusieurs façons de traiter la résolution du système MDA et les deux sections qui suivent décrivent les stratégies FIO et DAO employées à cet effet dans ce projet.

3.3.1 FIO

Dans la stratégie FIO (« Fully Integrated Optimization »), également connue sous l'appellation « Multidisciplinary Feasible » (MDF), on conserve le couplage entre les disciplines. La méthode consiste donc à appliquer une optimisation à un sytème pour lequel le couplage est résolu par une analyse multidisplinaire couplée (le système MDA). Le problème d'optimisation s'écrit alors comme suit :

$$\begin{aligned} &\min_{s,\,l_i} \sum_{i=1}^{N} F_i\left(s, r_i^*\right) \\ &\text{soumis à : } C_i\left(s, l_i, r_i^*\right) \geq 0 \quad i = 1..N \end{aligned} \tag{3.4}$$

La figure 3.2 illustre l'interaction entre l'optimiseur et le système MDA pour la stratégie FIO. La boite nommée MDA résout le couplage entre les disciplines. Son contenu est similaire au figures 3.1 et 4.8. L'optimiseur envoie donc les valeurs des variables au système MDA qui résout le couplage et retourne les valeurs de sortie r^*.

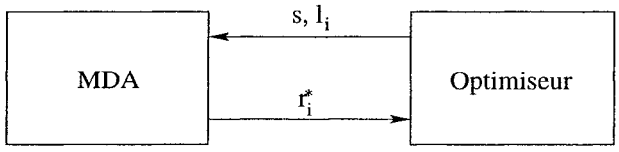

Figure 3.2 Interaction entre l'optimiseur et le système MDA pour la stratégie FIO

Dans le cas présent, $i = \{SOAPP, P1242, RD\}$, il n'y a pas de C_{SOAPP} (voir le

tableau 2.8). Le problème d'optimisation FIO peut donc s'écrire :

$$\begin{aligned} &\min_{\substack{s,\, l_{SOAPP},\\ l_{P1242},\, l_{RD}}} r^*_{SOAPP_{sfc}} \\ &\text{soumis à : } C_{P1242}\left(s, l_{P1242}, r^*_{P1242}\right) \geq 0 \\ &\qquad\qquad C_{RD}\left(s, l_{RD}, r^*_{RD}\right) \geq 0 \end{aligned} \tag{3.5}$$

Comme on le voit, les valeurs de sorties r^*_{SOAPP}, r^*_{P1242} et r^*_{RD} sont requises par la stratégie FIO. Il faudra donc résoudre le système MDA à chaque évaluation de l'optimiseur afin d'obtenir les bonnes valeurs des r^*_i.

3.3.2 DAO

La stratégie DAO (« Distributed Analysis Optimization »), aussi nommée « Individual Feasible » (IDF) [15], propose de relaxer le couplage afin de ne pas avoir à le résoudre à chaque fois que l'optimiseur lance le solveur. Dans ce but, on ajoute des variables intermédiaires au système, variables que l'on nomme t_i. En fait, les vecteurs t_i sont des copies des vecteurs r_i. La différence est que les vecteurs p_i sont maintenant constitués de composantes des t_j au lieu de composantes des r_j et que l'optimiseur peut faire varier les t_j. Ceci est illustré à la figure 3.3. Bref, les paramètres sont remplacés par des variables. Ceci vient donc relaxer le couplage. Toutefois, afin de s'assurer que la solution finale respecte le couplage entre les disciplines, il faut imposer de nouvelles contraintes en sortie, contraintes du type $t_i - r_i = 0$. Le problème d'optimisation s'écrit alors comme suit :

$$\begin{aligned} &\min_{s,l_i,t_i} \sum_{i=1}^{N} F_i\left(s, t_i\right) \\ &\text{soumis à : } C_i\left(s, l_i, r_i\left(s, l_i, t_j\right)\right) \geq 0 \quad i = 1..N, j = 1..N, j \neq i \\ &\qquad\qquad H_i = t_i - r_i\left(s, l_i, t_j\right) = 0 \end{aligned} \tag{3.6}$$

La figure 3.4 illustre l'interaction entre l'optimiseur et les différentes disciplines

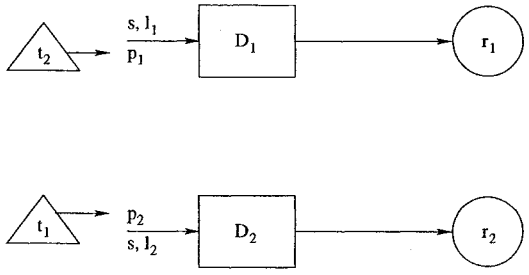

Figure 3.3 Découplage de 2 disciplines

pour la stratégie DAO. On y voit clairement que l'optimiseur envoi à chaque discipline les valeurs des variables qui lui sont propres et les disciplines retournent leur vecteur de sortie r. Le système MDA n'est donc pas résolu puisque aucune valeur de type r^* n'est requise.

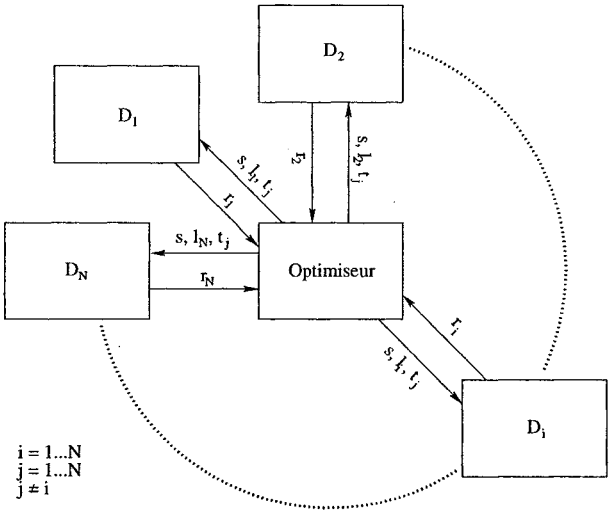

Figure 3.4 Interaction entre l'optimiseur et les disciplines pour la stratégie DAO

Les conditions sont ici les mêmes que pour FIO : $i = \{SOAPP, P1242, RD\}$. Il n'y a pas de dépendance de r_{P1242} ou r_{SOAPP} vis à vis de r_{RD}, alors il n'est pas nécessaire d'introduire une variables additionnelles t_{RD} et la contrainte de compatibilité associée $t_{RD} - r_{RD} = 0$. Comme précédemment, il n'y a pas de C_{SOAPP}. Le problème d'optimisation DAO peut donc s'écrire :

$$
\begin{aligned}
&\min_{\substack{s,l_{SOAPP},l_{P1242},l_{RD},\\ t_{SOAPP},t_{P1242}}} t_{SOAPP_{sfc}} \\
&\text{soumis à : } C_{P1242}\left(s, l_{P1242}, r_{P1242}\left(s, l_{P1242}, t_{SOAPP}\right)\right) \geq 0 \\
&\qquad C_{RD}\left(s, l_{RD}, r_{RD}\left(s, l_{RD}, t_{SOAPP}, t_{P1242}\right)\right) \geq 0 \\
&\qquad H_{SOAPP} = t_{SOAPP} - r_{SOAPP}(s, l_{SOAPP}, t_{P1242}) = 0 \\
&\qquad H_{P1242} = t_{P1242} - r_{P1242}(s, l_{P1242}, t_{SOAPP}) = 0
\end{aligned}
\tag{3.7}
$$

Il aurait été possible de minimiser $r_{SOAPP_{sfc}}$ à la place de $t_{SOAPP_{sfc}}$ mais ceci aurait compliqué la tâche de la minimisation de l'objectif par l'optimiseur. En effet, $t_{SOAPP_{sfc}}$ est une variable et non pas une valeur de sortie, donc n'est fonction d'aucune autre variable. Ceci linéarise la fonction à minimiser par l'optimiseur puisque la dérivée de l'objectif ($t_{SOAPP_{sfc}}$) par rapport à toutes les variables est nulle à l'exception de la dérivée par rapport à $t_{SOAPP_{sfc}}$ qui est égale à 1 par définition. Le choix de minimiser $t_{SOAPP_{sfc}}$ complique toutefois la tâche de satisfaire la contrainte de compatibilité $t_{SOAPP_{sfc}} - r_{SOAPP_{sfc}} = 0$ par l'optimiseur car $t_{SOAPP_{sfc}}$ sera contraint à avoir une valeur proche de $r_{SOAPP_{sfc}}$ tout au long de l'optimisation. En minimisant $r_{SOAPP_{sfc}}$, on obtiendrait l'effet contraire, c'est-à-dire que la fonction à minimiser par l'optimiseur serait plus complexe à résoudre.

3.4 Choix de la méthode d'optimisation

On peut séparer en deux grandes étapes un processus d'optimisation dans le cas des problème non-linéaires : la recherche globale et la recherche locale. La première permet de trouver les régions de l'espace de design où il y a un optimum potentiel. Le rôle de la recherche locale est ensuite de trouver l'optimum dans chacune de ces zones.

Dans le présent projet, on préfère se concentrer sur la recherche locale et on assume que la recherche globale est effectuée par les ingénieurs qui construisent le point

initial. Dans les faits, les fichiers initiaux pour chaque programme ont été modifiés jusqu'à ce qu'une solution réalisable soit trouvée. Cette restriction aux minimums locaux permet d'utiliser les méthodes à gradients, qui sont efficaces pour ce genre de problème. L'exploration de l'ensemble de l'espace de design par des méthodes d'explorations telles que les méthodes génétiques, de recuit simulé (« simulated annealing ») et autres pourra être entreprise dans un travail subséquent.

Comme on le voit au tableau 2.9, le nombre de variable pour les problèmes FIO et DAO réduits n'est pas très élevé : 34 pour FIO et 66 pour DAO. Par contre, il y a beaucoup de contraintes en sortie. Puisque l'on part d'un point faisable, qu'il n'y a pas trop de variables et qu'elles sont toutes réelles, on estime que les méthodes à gradients de type SQP seront efficaces. En effet, les méthodes SQP sont supérieures aux autres méthodes lorsque les hypothèses suivantes sont satisfaites : l'espace de design ne doit pas être trop vaste, les fonctions et les gradients doivent pouvoir être évalués avec suffisamment de précision et le problème ne doit pas être trop bruité. La première condition n'est pas assurément remplie pour DAO. Ce projet permettra donc de vérifier l'efficacité des méthodes SQP sur des espaces de design moyennement grand. Les deux dernières conditions devront être évaluées avec soin au cours du projet. En effet, il sera important de vérifier que la précision des programmes d'analyse est satisfaisante et que le bruit est suffisamment petit pour obtenir des gradients de qualité. Une étude de sensibilité des programmes d'analyse est présentée à l'annexe III comme base à cette évaluation de la précision des programmes.

Il y a une autre raison pour laquelle les méthodes à gradients sont préconisées : la stratégie DAO est formulée de telle sorte qu'elle soit efficace pour des méthodes à gradients. En effet, dans la méthodologie DAO, le problème est relaxé en introduisant des contraintes de compatibilité. Par conséquent, la méthode d'optimisation doit savoir comment se comportent ces contraintes par rapport aux variables pour faire

varier celles-ci de façon efficace. Ceci ne veut pas dire que les autres méthodes n'ont pas de chance d'être efficace. Par contre, avec une méthode à gradients, on estime pouvoir obtenir de meilleurs résultats. Cette affirmation reste à démontrer.

Il demeure cependant que les fonctions étudiés dans ce travail sont non-linéaires et présentent probablement une multitude de minimum locaux. Ainsi, il faut garder à l'esprit que ces méthodes ne fournissent qu'un moyen pour améliorer un design de départ, en minimisant la fonction jusqu'à un minimum local. Cette limitation est le prix à payer pour la rapidité à trouver un minimum.

Il y a plusieurs méthodes à gradients implantées dans le logiciel commercial iSIGHT. Celle qui a été retenue est une méthode SQP nommée NLPQL. Elle a été choisie car iSIGHT permet à l'usager de fournir les gradients pour cette dernière d'après les spécifications du guide de l'utilisateur. Ceci ne devrait pas être un avantage puisque l'on n'a pas d'expressions analytiques pour les dérivées mais il s'avère qu'il est beaucoup plus rapide de calculer tous les gradients à l'extérieur d'iSIGHT [30]. En effet, ceci permettrait de diminuer de beaucoup les opérations de lecture et d'écriture effectuées par iSIGHT et comme celles-ci sont lentes, elles ralentissent le processus de façon notable. Malheureusement, dû à un problème dans le code d'iSIGHT, cette fonctionnalité n'est pas au point et n'a, par conséquent, pu être utilisée dans le présent projet. On a tout de même gardé NLPQL pour son efficacité à résoudre le problème étudié et ainsi, lorsque que les développeurs de iSIGHT auront réglé le problème, il sera facile de faire des optimisations beaucoup plus rapidement avec NLPQL qu'avec les autres techniques. Une description de NLPQL est donnée à l'annexe II.

3.5 Technique de réduction de pas

Comme il a été mentionné précédemment, les méthodes SQP nécessitent la connaissance des gradients de la fonction coût et des contraintes à chaque itéra-

tion. Dans le cas présent, ceux-ci sont obtenus par différences finies car les relations analytiques entre les variables et la fonction coût et les contraintes sont inconnues. iSIGHT définit comme suit le calcul des gradients pour NLPQL :

$$\frac{\partial F}{\partial x} = \frac{F(x+hx) - F(x)}{hx} + O(x) \tag{3.8}$$

où h est commun à toutes les variables et $O(x)$ est l'erreur de troncature à l'ordre un pour l'évaluation de la dérivée partielle $\frac{\partial F}{\partial x}$ par différence finie amont.

La sélection d'un intervalle pour le pas de différenciation est critique. En effet, un pas hx trop grand donne des gradients imprécis et un pas trop petit est affecté par le bruit numérique des programmes d'analyse ainsi que par les erreurs d'arrondi. Il est donc nécessaire de trouver un compromis permettant de s'affranchir du bruit numérique tout en ayant une bonne précision sur les gradients. Une technique mentionnée par Papalambros [29] permet selon cet auteur de trouver un pas « idéal » pour toutes les variables [1]. Par contre, étant donnée la taille du problème et la variation du bruit d'une variable à l'autre (voir l'annexe III), une méthode plus empirique a été utilisée dans ce projet. En effet, la technique proposée par Papalambros demande beaucoup de travail de calcul préliminaire et n'est pas garantie de succès, notamment pour des problèmes de grande taille.

Une étude de sensibilité a d'abord permis de déterminer un intervalle global pour le pas ($[h_{min_{global}}, h_{max_{global}}]$) à l'intérieur duquel les gradients sont suffisamment précis (voir l'annexe III). Basé sur cet intervalle, des optimisations ont été conduites afin de déterminer un intervalle $[h_{min}, h_{max}]$ pour FIO et DAO permettant d'atteindre rapidement le meilleur minimum connu. Puisque le bruit borne la valeur de h_{min}, il est

[1] Cette technique permet de déterminer le pas de différenciation qui est le compromis optimal entre l'erreur de troncature et l'erreur liée à la présence de bruit lors de l'évaluation de $F(x+hx)$ et $F(x)$.

difficile d'obtenir des gradients de qualité près de l'optimum. La figure 3.5 illustre ce phénomène. Comme on le voit, le minimum est mal défini à cause du bruit. De plus, le pas de différenciation h_{min} doit être plus grand que le pas caractéristique du bruit pour éviter que l'optimiseur reste coincé dans un optimum dû au bruit. Il est donc impossible d'atteindre avec une grande précision le minimum de la fonction bruitée. Mathématiquement, les gradients près du minimum sont imprécis et il est difficile pour l'optimiseur de trouver un point minimum qui satisfait le critère de convergence de premier ordre de Karush-Kuhn-Tucker (KKT).

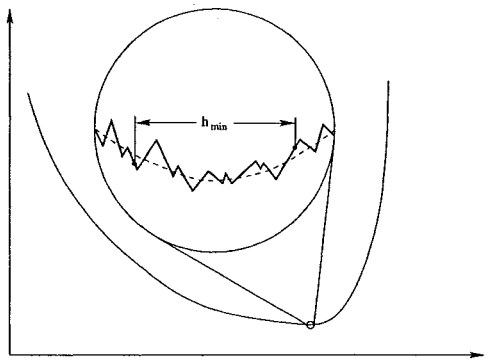

Figure 3.5 Dualité précision-bruit versus h_{min} à l'optimum

La technique de réduction de pas consiste à faire une première optimisation avec le pas h_{max}. On effectue ensuite d'autres optimisations en réduisant le pas jusqu'au pas h_{min}. L'utilisation d'un pas plus grand au début du processus d'optimisation permet de s'approcher plus rapidement du minimum et peut même permettre à l'optimiseur de franchir des minimums locaux dans lesquels il serait resté coincé avec un pas plus petit. L'utilisation de pas plus petits pour les optimisation subséquentes permet d'améliorer la précision des gradients et facilite ainsi la convergence vers le minimum.

Il n'est pas garanti que la méthode de réduction de pas donne des meilleurs résultats qu'une optimisation avec un pas unique soigneusement choisi. On sait qu'il existe un minimum dans la zone exploitée par l'optimiseur et que les pas compris

dans l'intervalle $[h_{min}, h_{max}]$ permettent à l'optimiseur de se diriger dans la zone où se trouve ce minimum à moins que l'optimiseur ne reste coincé dans un minimum local dû au bruit. La technique de réduction de pas permet alors d'assurer une convergence rapide vers la zone où se trouve le minimum sans avoir à tester plusieurs pas.

3.6 Optimisation avec la stratégie FIO

On effectue une optimisation avec pour seul objectif de minimiser le *sfc*. Le problème possède 34 variables, 112 contraintes et 1 objectif (voir le tableau 2.9). Ces derniers appartiennent respectivement aux ensembles s, l_i, C_i et F_{SOAPP} du tableau 2.8 où $i = \{SOAPP,\ P1242,\ RD\}$. Il faut noter que toutes les variables des ensembles de type t sont des paramètres avec FIO car ce sont des variables couplées et que le couplage est géré par une enveloppe globale pour FIO (voir la sous-section 4.1.9).

Tableau 3.2 Description du plan d'optimisation pour FIO

	$NLPQL_1$	$NLPQL_2$
Nombre d'itérations maximum	15	15
Pas de différentiation	0.005	0.0045
Pas minimum de différentiation	0.002	0.002
Précision	0.001	0.001
Type de différentiation	avant	avant
Tolérance sur les contraintes d'inégalités	0.001	0.001

La méthode de réduction de pas comporte ici deux niveaux, tel que montré au tableau 3.2, et l'intervalle $[h_{min}, h_{max}] = [0.0045, 0.005]$. Il est malheureusement impossible de réduire beaucoup h_{min} car les programmes sont quelque peu bruités (voir l'annexe III). Un pas de différenciation inférieur à 0.0045 est inefficace tout comme un pas supérieur à 0.005 selon les tests effectués avec iSIGHT. L'optimiseur n'a donc pas beaucoup de chance de descendre vraiment jusqu'à l'optimum car celui-ci est mal défini à cause du bruit. De plus, une fois à proximité du minimum, le pas sera trop grand pour obtenir les gradients nécessaires à une diminution supplémentaire de l'objectif.

Les conséquences de ceci ne sont pas très graves car de toutes façons une précision inférieure à 0.1% n'est pas significative à ce stade du design. C'est pourquoi la précision et la tolérance sur les contraintes d'inégalité sont ajustées à 0.001. Ceci permettra à NLPQL d'arrêter sa recherche lorsque cette précision sera atteinte. Un nombre maximal d'itérations est tout de même imposé pour chaque étape. Ces nombres ont été choisis de façon à minimiser le temps d'optimisation sans compromettre le résultat.

3.7 Optimisation avec la stratégie DAO

Avec la méthode DAO, l'objectif de l'optimisation est de nouveau de minimiser le *sfc*. On a alors 66 variables, 144 contraintes et 1 objectif (voir le tableau 2.9). Ils appartiennent aux ensembles s, l_i, t_i, C_i et F_{SOAPP} du tableau 2.8 où $i = \{SOAPP,\ P1242,\ RD\}$.

Tableau 3.3 Description du plan d'optimisation pour DAO

	$NLPQL_1$	$NLPQL_2$	$NLPQL_3$
Nombre d'itérations maximum	15	15	15
Pas de différenciation	0.005	0.0045	0.0045
Pas minimum de différenciation	0.002	0.002	0.002
Précision	0.001	0.001	0.001
Type de différenciation	avant	avant	avant
Tolérance sur les contraintes d'égalités	0.1	0.01	0.001
Tolérance sur les contraintes d'inégalités	0.01	0.00316*	0.001

* On veut des intervalles égaux alors $\frac{a}{x} = 0.001 \wedge ax = 0.01 \Rightarrow a = \sqrt{0.00001} = 0.00316$

L'intervalle $[h_{min}, h_{max}]$ est le même que pour FIO. On effectue trois étapes NLPQL, tel que montré au tableau 3.3, pour des raisons énoncées plus loin. Les deux dernières optimisations ont le même pas de différenciation car il a été jugé in-

utile de faire une réduction de pas en trois niveaux compte tenue de l'étroitesse de l'intervalle $[h_{min}, h_{max}]$.

Les autres réglages de NLPQL sont faits en suivant la même logique que pour FIO à l'exception des critères de tolérance sur la violation des contraintes et des poids sur les contraintes de compatibilité. Le but de tout ceci est d'accélérer l'optimisation et d'obtenir un meilleur minimum. En effet, lors des premiers tests d'optimisation avec DAO, il a été observé que l'optimiseur n'arrivait pas à réduire la fonction coût. Pour comprendre la cause de ce problème, il faut comprendre le fonctionnement de NLPQL. Cette méthode commence par trouver une direction de descente et par la suite elle trouve une distance de descente dans cette direction. La distance de descente est basée sur la minimisation d'une fonction coût qui tient compte de la violation des contraintes. On comprend donc que si l'optimiseur n'arrive pas à descendre vers un minimum à partir d'un point initial, c'est que la distance optimale de descente trouvée est toujours nulle. Cette situation doit se produire seulement lorsqu'un minimum est atteint. Or le point de départ n'est pas un minimum. Afin de réussir à descendre, les contraintes ne doivent pas être violées. NLPQL permet cependant de définir un intervalle de tolérance sur la satisfaction des contraintes. Il suffit alors de donner une valeur de tolérance plus grande au début de l'optimisation afin de permettre à l'optimiseur de descendre vers le minimum en violant quelques contraintes. On réduit ensuite cette tolérance pour NLPQL_2 et NLPQL_3 afin de satisfaire les contraintes avec la tolérance de design à la fin. C'est pourquoi la tolérance sur les contraintes d'égalité et d'inégalités varient d'une étape de NLPQL à l'autre.

On varie les poids des contraintes de compatibilité pour les mêmes raisons. Au départ, les poids sont faibles et donc influencent moins la distance de descente trouvée par NLPQL. Pour NLPQL_2 et NLPQL_3, on augmente le poids sur les contraintes de compatibilité forçant ainsi NLPQL à les prendre en compte. On ajuste les poids pour

$NLPQL_3$ afin que les contraintes de compatibilités soient satisfaites aussi sévèrement qu'avec FIO. On s'assure ainsi que les formulations FIO et DAO sont équitables pour NLPQL.

CHAPITRE 4

INTÉGRATION

Les chapitres précédents ont portés sur la définition d'un modèle mathématique du problème. Ce chapitre décrit la méthodologie utilisée pour intégrer et appliquer ce modèle. La première étape de ce processus consistera à établir la communication entre les différents programmes d'analyse par l'intermédiaire d'un processus d'intégration. On procédera ensuite à la description de l'intégration des modèles FIO et DAO dans le logiciel iSIGHT.

4.1 Intégration des programmes

Comme il a été mentionné aux chapitres précédents, il y a un couplage entre les différents programmes, c'est-à-dire que les sorties de certains programmes sont nécessaires en entrée d'autres programmes. Ceci implique que les programmes doivent pouvoir communiquer entre eux afin de résoudre ce couplage. De plus, iSIGHT doit pouvoir interagir avec les différents programmes d'analyse afin de réaliser l'optimisation. Il est donc nécessaire d'établir un moyen de communication entre l'ensemble des applications impliquées dans le processus d'optimisation.

4.1.1 Choix d'une méthode d'intégration

Deux moyens principaux ont été envisagés pour effectuer l'intégration. Le premier consiste à utiliser directement le module d'intégration du logiciel iSIGHT afin d'établir la communication entre les différents programmes. Avec cette méthode, le modèle serait complètement défini dans iSIGHT. Le second moyen consiste à enrober chaque programme à l'aide d'enveloppes (« wrappers » en anglais), écrits dans un langage de programmation neutre, afin que chacun d'entre eux soit habilité à communiquer

avec un fichier commun à tous. On utilisera alors le module d'intégration du logiciel iSIGHT pour établir la communication entre ce fichier unique et iSIGHT. Avec cette méthode, une bonne partie du modèle est intégré à l'extérieur de iSIGHT. Ce dernier est ainsi utilisé uniquement pour l'optimisation.

La figure 4.1 illustre les différences entre les deux méthodes d'intégration. On voit ainsi clairement le couplage entre les disciplines pour la méthode 1. Pour sa part, la méthode 2 centralise l'information et le couplage sera pris en charge par un gestionnaire habileté à exécuter les différentes enveloppes. On remarque immédiatement qu'il est beaucoup plus simple d'établir la communication entre les différents programmes avec la seconde méthode. Par contre, le prix à payer est la programmation d'enveloppes.

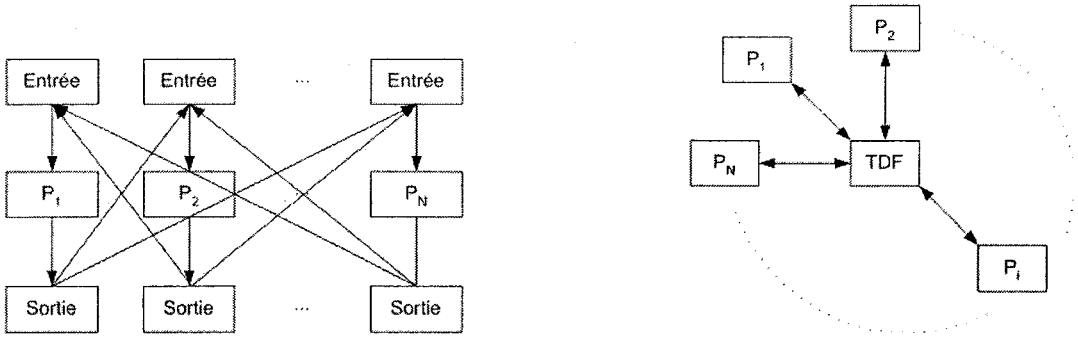

Figure 4.1 Comparaison des deux méthodes d'intégration

La deuxième méthode d'intégration a été choisie. Cette décision est basée principalement sur deux points :

1. La première méthode avait été utilisée dans un projet antérieur (IDOT) et les résultats avaient démontrés que les fichiers Mdol-Tcl deviennent rapidement complexes et difficiles à débugguer et à maintenir ;
2. La deuxième méthode semble présenter plus de souplesse lorsqu'il faut faire l'intégration de problèmes complexes dans un logiciel comme iSIGHT. Le présent travail permettra de valider cette hypothèse.

4.1.2 Choix du langage

Le langage utilisé pour effectuer l'enrobage des programmes est le Perl. Ce langage est tout à fait approprié pour le type de tâches qui consistent en grande majorité à manipuler des données dans des fichiers ASCII. Perl permet en général de faire en quelques lignes des opérations que des langages comme le C++ font en des dizaines de lignes. Son seul défaut est d'être un langage interprété, donc relativement lent. Il est toutefois possible de transformer un code en langage Perl en code en langage C pour ensuite compiler ce code C. Cette option est encore au stade de recherche et permettra dans le futur d'obtenir des programmes Perl qui s'exécutent aussi vite qu'un programme en C [31]. De plus, Perl est indépendant de la plateforme utilisée (UNIX ou PC).

Dans ce projet, les enveloppes sont donc écrites en Perl et tous les programmes d'analyse deviennent ainsi des programmes en Perl communicant avec un seul fichier.

4.1.3 Fichier commun

Le fichier commun utilisé par les enveloppes est nommé TDF, pour « Turbine Data File ». C'est un simple fichier texte qui contient le nom et la valeur de chaque variable. Le fichier est divisé par discipline et pour chacune d'elles, il y a une section entrée et une section sortie. Pour le cas présent, il y a donc une partie performance, une partie fluide et une partie structure rotative. Certaines parties connexes sont également présentes même si aucun programme ne leur est associé. On retrouve ainsi les structures statiques et le refroidissement. Chaque variable est définie par son nom suivi de := et de sa valeur.

On obtient ainsi un fichier central contenant l'information relative à tous les programmes. C'est uniquement avec ce fichier qu'iSIGHT interagira.

4.1.4 Fonctionnement des enveloppes

Une enveloppe est une programme, dans le cas présent écrit en langage Perl qui enveloppe chaque programme d'analyse et ses fichiers d'entrée et de sortie. Ce programme (enveloppe) est habilité à communiquer avec un fichier commun, le TDF. L'enveloppe est capable de générer les fichiers d'entrées à partir du TDF et de mettre à jour le TDF à partir des fichiers de sorties du programme qu'il enrobe. La figure 4.2 illustre le fait que l'enveloppe devient équivalent à une boîte noire qui inclut le programme d'analyse. On y observe que toute enveloppe prend simplement le fichier commun (TDF) en entrée et donne une mise à jour de ce dernier en sortie.

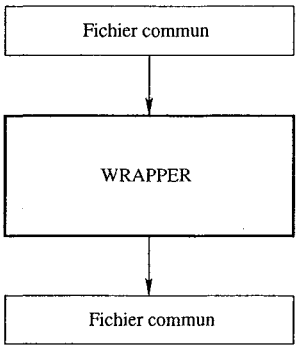

Figure 4.2 L'enveloppe, une boîte noire

La figure 4.3 illustre en détail les opérations faites à l'intérieur d'une enveloppe lorsqu'on l'exécute. L'enveloppe correspond à la partie dans l'encadré en pointillé. La première étape consiste à générer un fichier d'entrée du programme. Pour ce faire, la manière la plus simple est de copier un fichier d'entrée de référence (« template ») et de venir y mettre à jour les valeurs en se servant des valeurs retrouvées dans le fichier TDF. Une fois cette étape franchie, l'enveloppe lance l'exécution du programme d'analyse et capte les messages d'erreurs. Ensuite, le ou les fichiers de sortie sont lus et l'enveloppe vérifie qu'ils sont conformes. Si tel est le cas, le fichier commun (TDF) est mis à jour avec les nouvelles valeurs ; sinon un message d'erreur est retourné et l'enveloppe échoue. Ceci permet à iSIGHT de savoir qu'il y a un problème. Ce dernier

a ainsi la possibilité de gérer ces échecs.

Toutes les enveloppes sont des entités complètement indépendantes les unes des autres et ils ont pour unique point commun le TDF. Ceci permet de les assembler de n'importe quelle façon très facilement.

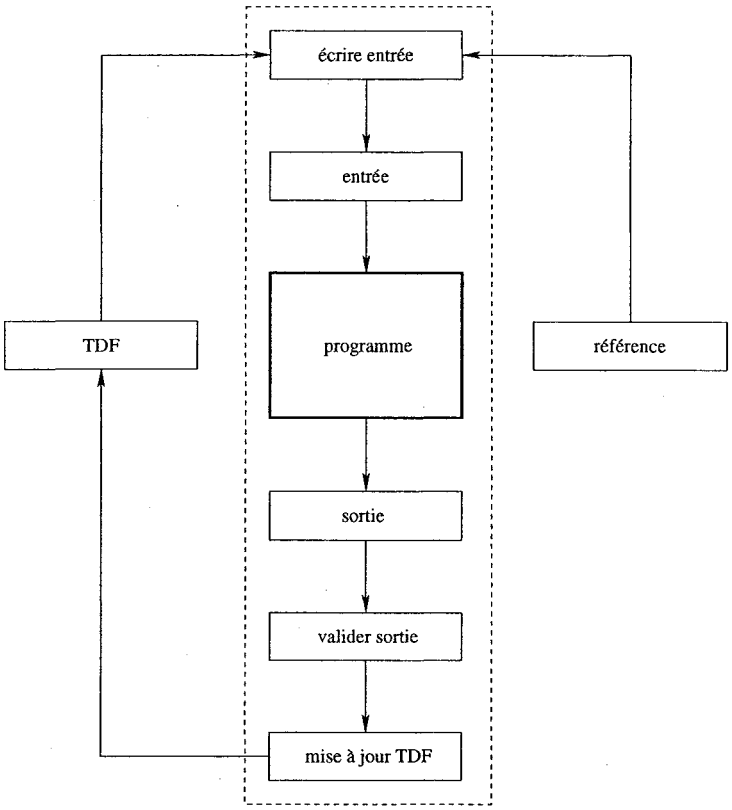

Figure 4.3 Schéma détaillé d'une enveloppe

4.1.5 Enveloppe de SOAPP

L'enveloppe de SOAPP est le plus simple. Il ressemble en tout point à l'enveloppe présenté à la figure 4.3. Il ne fait que générer un fichier d'entrée, exécuter SOAPP et mettre à jour le TDF comme illustré à la figure 4.4.

La principale difficulté avec SOAPP est le traitement des fichiers d'entrée et de

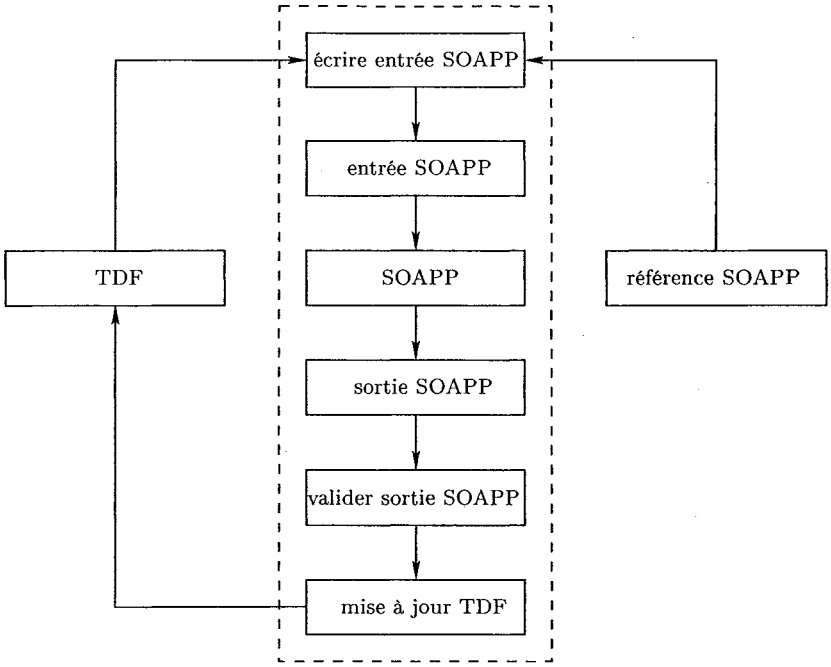

Figure 4.4 Schéma détaillé de l'enveloppe de SOAPP

sortie. En effet, ces fichiers peuvent être de formes diverses selon le moteur. Dans le cas présent, on traite seulement les fichiers propres au moteur étudié. Notons cependant qu'il serait complexe décrire un « parser » général pour SOAPP (voir la section 2.1.1).

4.1.6 Enveloppe de P1242

L'enveloppe de P1242 est un peu plus complexe que celui de SOAPP car P1242 est couplé avec lui-même. En effet, la *Reaction* et la perte entre les stations 5 et 6, nommé Y_{56}, sont à la fois des entrées et des sorties de P1242. L'enveloppe comporte donc une boucle telle qu'illustrée à la figure 4.5. En fait la *Reaction* et Y_{56} sont des fonctions des sorties de P1242 et c'est pour cette raison qu'il faut les faire converger. Ces fonctions sont définies comme suit :

$$\begin{aligned} Reaction &= Reaction_{optimal}(PR) - \Delta Reaction \\ Y_{56} &= Y_{56}(R_1, R_2, A_1, A_2, AL_{56}, \alpha) \end{aligned} \tag{4.1}$$

où

$PR =$ Ratio de pression de l'étage de turbine (p_0/p_5)

$R_1 =$ Rayon moyen à la station 5

$R_2 =$ Rayon moyen à la station 6

$A_1 =$ Aire de passage à la station 5

$A_2 =$ Aire de passage à la station 6

$AL_{56} =$ Longueur du conduit entre les stations 5 et 6

$\alpha =$ Angle absolu de l'écoulement à la ligne moyenne de la station 5

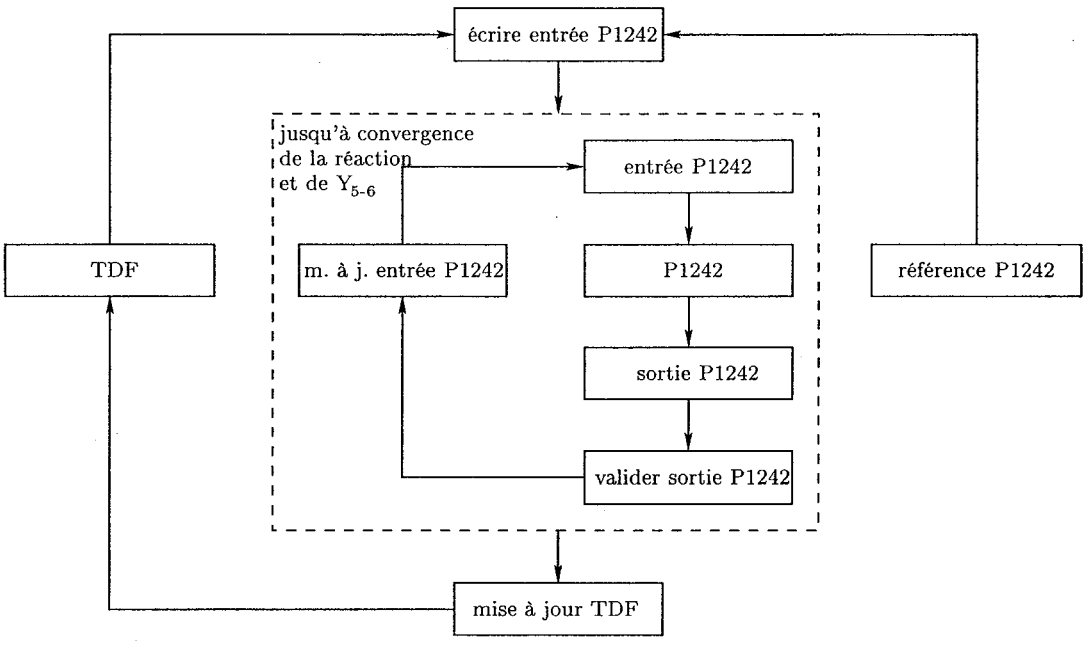

Figure 4.5 Schéma détaillé de l'enveloppe de P1242

Le couplage n'est pas très fort et il est habituellement résolu en deux itérations (deux exécutions de P1242). Aucun cas n'a été répertorié où plus de quatre itérations ont été nécessaires à la résolution du couplage. Un paramètre limite tout de même le nombre d'itération à 15 pour éviter que l'enveloppe de P1242 puisse exécuter une boucle infinie.

4.1.7 Enveloppe de RotorDesigner

Puisque RD calcule un seul étage à la fois, son enveloppe a la propriété de l'exécuter autant de fois qu'il y a d'étages. De plus, il peut arriver qu'une analyse plastique en 2-D soit nécessaire (voir la sous-section 2.1.3). Pour ce faire, l'enveloppe de RD doit alors exécuter l'enveloppe de P0831.

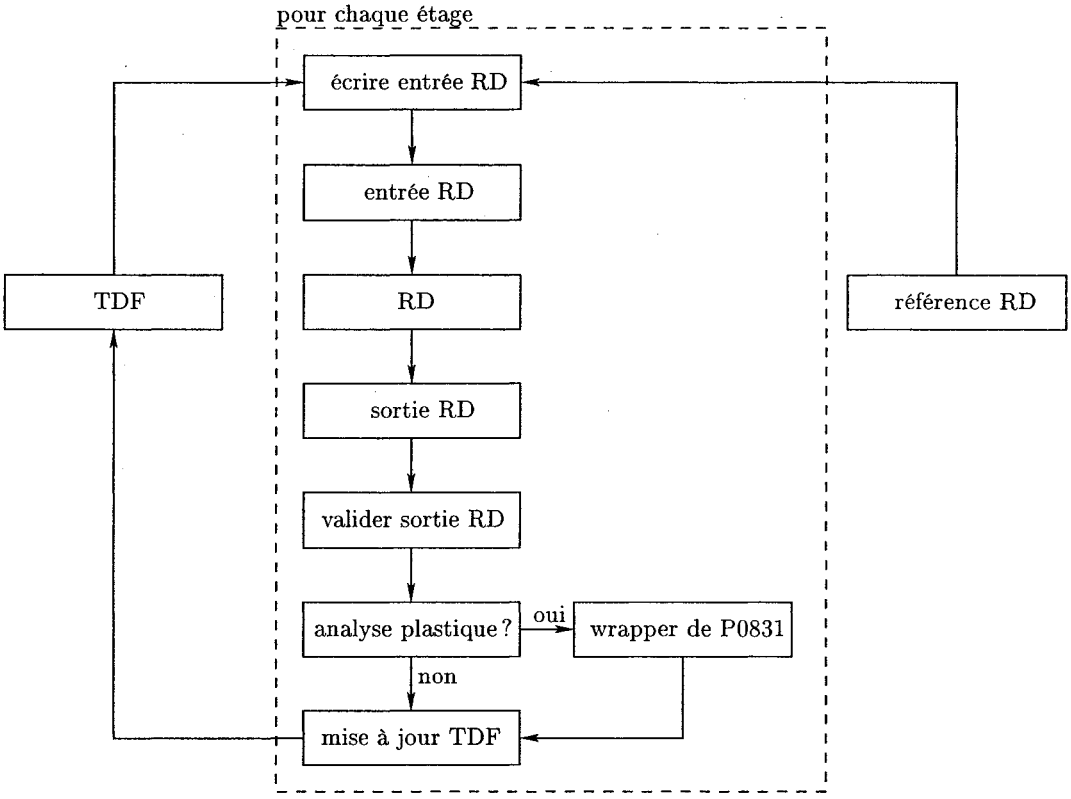

Figure 4.6 Schéma détaillé de l'enveloppe de RD

L'enveloppe de RD exécute donc séquentiellement RD pour chaque étage de turbine en plus d'exécuter l'enveloppe de P0831 au besoin. On considère ainsi P0831 comme une partie intégrante de l'enveloppe de RD. Ceci évite d'avoir à intégrer P0831 dans iSIGHT. On exécute P0831 si la marge de résistance à la survitesse du disque est inférieure ou près de la valeur critique uniquement, car il est inutile de lancer un calcul en 2-D pour avoir plus de précision si tel n'est pas le cas.

Puisque chaque étage est indépendant, il serait possible de modifier l'enveloppe de RD afin de permettre une exécution en parallèle pour chaque étage mais ceci n'a pas été fait dans ce projet.

4.1.8 Enveloppe de P0831

L'enveloppe de P0831 est constitué d'une séquence de deux appels à P0831 (voir la figure 4.7). Le premier appel permet de faire une analyse élastique. Cette dernière permet de fournir les données nécessaires à l'analyse plastique.

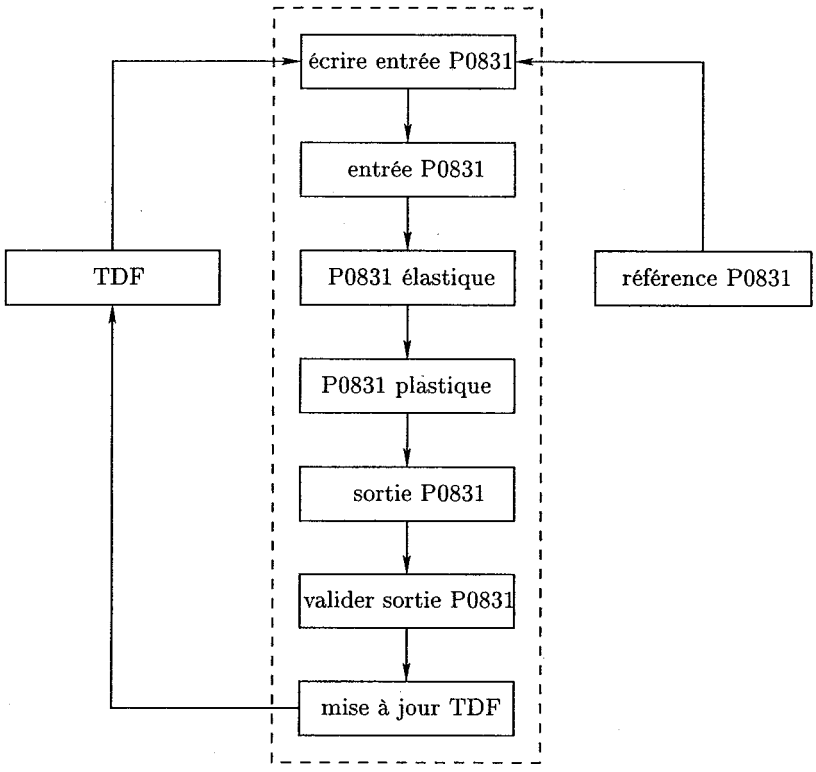

Figure 4.7 Schéma détaillé de l'enveloppe de P0831

4.1.9 Enveloppe global

La stratégie FIO consiste à optimiser le système couplé (voir la sous-section 3.3.1). Dans ce cas, on devra donner à iSIGHT une seule enveloppe qui contient tous les autres. Cette enveloppe global devra résoudre le couplage entre les différents programmes en plus d'appeler ces derniers dans l'ordre approprié. Puisque les enveloppes sont des modules indépendants, il est très facile de les assembler.

Pour le cas présent, il y a un couplage entre SOAPP et P1242 et ce dernier doit être résolu avant que RD soit exécuté. Une première enveloppe ayant pour tâche de résoudre le couplage est donc créé. Ce faisant, SOAPP et P1242 forment une enveloppe unique. L'enveloppe de RD doit être exécuté à la fin car il nécessite de connaître, entre autres, des températures calculées par SOAPP et P1242. La figure 4.8 illustre l'enveloppe global.

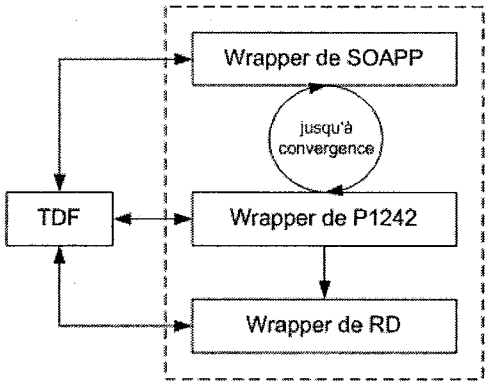

Figure 4.8 Schéma détaillé de l'enveloppe global

C'est cette enveloppe qui devient le programme donné à iSIGHT pour la stratégie FIO. Ce dernier le voit comme une boîte noire et interagit avec lui uniquement par l'intermédiaire du TDF.

4.2 Intégration dans iSIGHT

Afin d'utiliser iSIGHT, il faut bien sûr y intégrer les enveloppes. On présente dans un premier temps une description d'iSIGHT. Les intégrations de FIO et DAO dans iSIGHT suivront.

4.2.1 Description de iSIGHT

iSIGHT[1] est un logiciel commercial proposant un environnement convivial pour l'intégration et l'optimisation. Le logiciel possède toutes les fonctionnalités nécessaires pour lire et écrire dans des fichiers de format texte et exécuter tout type de programme sur la plupart des plateformes standard (PC, IBM, HP, SGI, etc). Il fonctionne tant sous MS Windows que sous les systèmes de type UNIX (Linux, AIX, HP-UX, IRIX, etc). On peut y intégrer autant de programme d'analyse que l'on veut et les faire interagir entre eux par l'utilisation d'opérateurs logiques ou de boucles.

iSIGHT offre également des fonctionnalités permettant de faire des « design of experiment » (DOE), des simulations de type Monte Carlo, des analyses six sigma (6σ) et il possède plusieurs autres attributs et fonctions qui en font un outil très puissant pour l'exploration du design et l'optimisation en ingénierie.

Le langage interne de iSIGHT s'appelle le Mdol. Le coeur de iSIGHT est toutefois construit en Tcl avec une interface graphique en Tk. Tcl est un langage supporté et augmenté par iSIGHT et qui permet de faire ce que le Mdol ne peut, ou ne veut, ou ne doit pas faire. Le langage Tcl est interprété et non pas compilé. Il est donc relativement lent, ce qui cause certains problèmes lorsque les programmes à exécuter sont rapides. En effet, lorsque tel est le cas, le temps mis par iSIGHT pour effectuer des opérations internes est une fraction importante du temps total d'optimisation.

[1] www.engineous.com

Dans ce projet, les programmes sont très rapides et donc iSIGHT n'est pas l'outil le mieux adapté si on prend seulement en considération l'aspect efficacité.

Par contre, iSIGHT a été choisi par P&WC pour ce projet en raison de sa grande polyvalence et des diverses méthodes qu'il offre de façon intégrée. En fait, il n'y a pas que l'efficacité de l'optimisation qui soit en jeu. Les ingénieurs pourraient vouloir par exemple utiliser les fichiers iSIGHT générés lors de ce projet pour faire des simulations de types Monte Carlo ou autres. Le fait d'avoir utilisé iSIGHT ouvre donc des perspectives intéressantes pour la suite du projet.

Le fichier d'entrée du logiciel iSIGHT est le « description file », qui contient toutes les informations nécessaires pour définir un problème spécifique. L'unité syntaxique de base d'un « desciption file » est nommé le « task » et on y définit tous les paramètres, le « parsing » des fichiers d'entrée et de sortie, les exécutions des programmes d'analyse ainsi que les options liées à l'optimisation. Un « task » peut contenir un nombre illimité de « sub-task » et dans ce cas, le « task » principal agit à titre de coordonnateur.

4.2.2 Étapes d'intégration d'un problème dans iSIGHT

Le but n'est pas ici de donner un cours d'introduction à iSIGHT mais plutôt de rendre compte de l'expérience acquise lors de ce projet tout en donnant un bref aperçu du travail à accomplir pour intégrer un problème dans iSIGHT.

Il y a trois étapes principales à suivre pour faire l'intégration rapidement :

1. Générer la version statique du « description file » de iSIGHT.
2. Insérer un « Perform Block » au début du fichier statique.
3. Générer la version dynamique du « description file ».

Dans un fichier dynamique, il est possible de définir, par exemple, qu'il y a n étages

de turbine tandis que dans un fichier statique, tout doit être défini explicitement, donc n doit être remplacé par une constante. La définition d'un fichier dynamique nécessite l'insertion d'un « Perform Block » au début du « description file » et dans lequel on attribut une valeur à n.

Le logiciel iSIGHT s'initialise uniquement à partir d'un fichier statique. Cependant, iSIGHT possède un pré-processeur qui transforme un fichier dynamique en fichier statique et permet ainsi l'initialisation de iSIGHT. Il est ainsi possible de définir un fichier dynamique pouvant s'adapter à des turbines avec des nombre d'étages différents et de le transformer de façon automatique, via le pré-processeur de iSIGHT, en un fichier statique spécifique à une turbine à gaz en particulier.

La figure 4.9 illustre le processus qui mène du fichier dynamique à l'initialisation de iSIGHT. Un pré-processeur transforme d'abord le fichier dynamique en fichier statique tel que mentionné précédemment. Le processeur Mdol interprète ensuite le fichier statique et initialise iSIGHT. Un exemple de passage du fichier dynamique au fichier statique via le pré-processeur de iSIGHT est donné à la figure 4.12, section 4.2.2.

Générer la version statique

Pour générer la version statique du « description file », il suffit de lancer iSIGHT et de créer un nouveau « task » en se servant de l'interface graphique (GUI). On peut ainsi définir toutes les variables, les contraintes, l'objectif, tous les « sub-tasks », les techniques d'optimisation, le « parsing » des fichiers d'entrée et de sortie, etc. Bref, on peut, à peu de chose près, définir le problème dans son ensemble.

Pour aller dans les détails de l'intégration, il faut ensuite éditer le « description file », généré à partir du GUI de iSIGHT, avec un éditeur de texte et le modifier. En

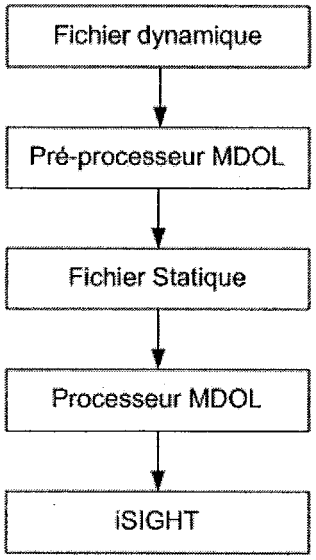

Figure 4.9 Étape menant du fichier dynamique à l'exécution de iSIGHT

effet, l'interface graphique de iSIGHT ne permet pas à l'usager d'utiliser toutes les fonctionnalités du langage Mdol-Tcl.

On édite donc le « description file » pour y insérer des « Tcl Blocks » permettant, par exemple, de venir initialiser automatiquement les facteurs d'échelle des variables ou le poids des contraintes. On fait ceci en se servant des « application programming interface (api) » qui sont des commandes de la version iSIGHT de Tcl. La figure 4.10 illustre la fonction des « api », qui consiste à permettre le transfert d'informations entre les deux langages employés dans iSIGHT, soit le Mdol et le Tcl.

Insertion du « Perform Block »

Pour créer le fichier dynamique, il faut que le nombre de turbines, d'étages de turbine, de lobes par attaches, etc (voir la section 2.2) soient des variables pour iSIGHT. On doit pouvoir donner une valeur à ces variables et ainsi permettre à iSIGHT de générer le fichier statique correspondant à une turbine spécifique. Pour ce faire, iSIGHT permet d'introduire un « Perform Block » dans le fichier tel que mentionné

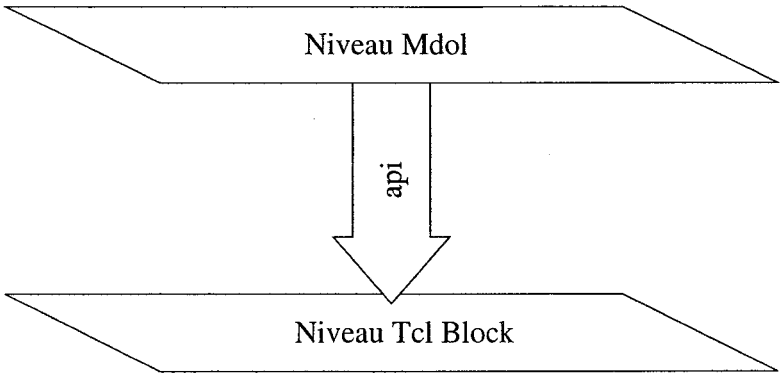

Figure 4.10 Fonction des « api »s

précédemment. Ce bloc est écrit en Tcl et on le place généralement au début du fichier. Il est exécuté par le pré-processeur de iSIGHT. Ceci permet d'aller chercher de l'information dans des fichiers pour permettre au pré-processeur de iSIGHT de générer le fichier statique et ainsi pouvoir s'initialiser.

Il est ici bien important de comprendre que ce qui est défini dans le « Perform Block » est complètement distinct du reste de l'information contenue dans le « description file ». L'information qu'il contient est accessible uniquement via des « Insert Blocks ». Il faut s'imaginer qu'il y a deux niveaux de variables dans un « description file » dynamique. Il y a le niveau Mdol-Tcl qui est le niveau interprété par le processeur de iSIGHT et le niveau Tcl définit dans le « Perform Block » et interprété par le pré-processeur de iSIGHT. Le pont entre ces deux mondes est le « Insert Block » qui permet de prendre de l'information au niveau Tcl et de l'amener au niveau Mdol-Tcl tel qu'illustré à la figure 4.11.

Générer la version dynamique

Comme il a été mentionné à la section précédente, il faut insérer des « Insert Blocks » dans le fichier afin de profiter de l'information acquise dans le « Perform Block ». C'est de cette façon qu'on pourra définir le « description file » dynamique.

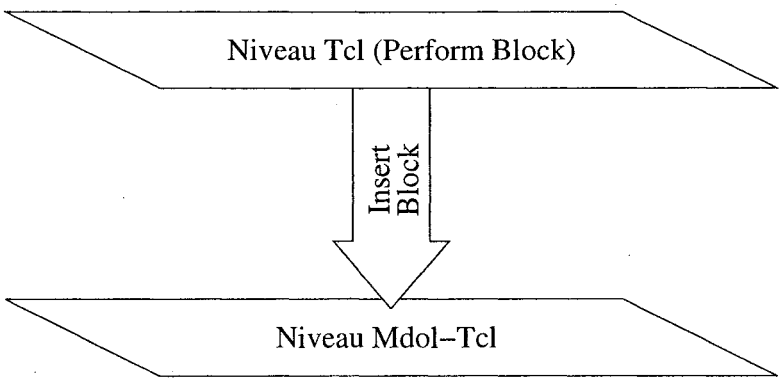

Figure 4.11 Les deux niveaux d'information de iSIGHT

Le fichier résultant pourra par conséquent s'adapter à différentes configurations de turbine. On évite ainsi d'avoir à créer un « description file » pour chaque turbine à étudier. Dans le présent projet, on étudie un seul moteur mais les « description file » associés aux stratégies FIO et DAO sont dynamiques pour le bénéfice des projets futurs.

On donne ici un exemple simple afin d'illustrer le processus qui mène du fichier dynamique au fichier statique (voir la figure 4.12). Dans cette exemple, on suppose que le « Perform Block » contient une variable nommée *nbetages* et on l'initialise à 2. On aurait tout aussi pu l'initialiser en allant lire une valeur dans un fichier. Un peu plus loin dans le fichier dynamique, on déclare une variable de sortie nommée *eta* et qui représente les rendements de chaque étage de turbine. Cette variable est donc un vecteur de dimension *nbetages*. On la définit donc à l'intérieur d'un « Insert Block » de manière à pouvoir accéder à la valeur de *nbetages* définie dans le « Perform Block ». Un fois que le fichier dynamique est interprété par le pré-processeur Mdol, le résultat est un fichier statique dans lequel la variable *eta* est un vecteur de dimension 2. On obtient ainsi un fichier statique propre à un moteur en particulier de façon automatique grâce au fichier dynamique et au pré-processeur Mdol.

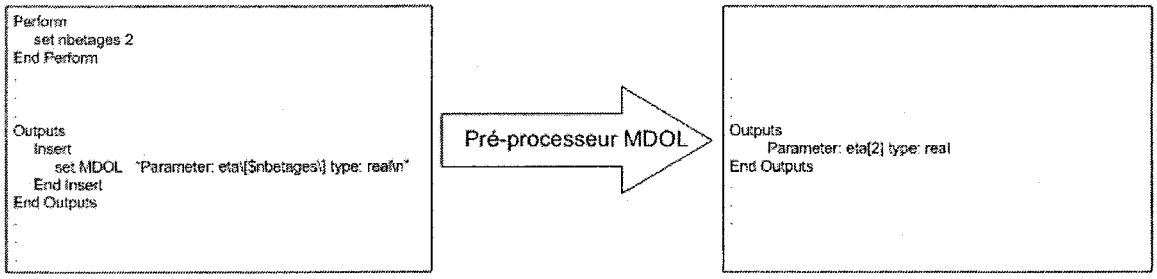

Figure 4.12 Exemple de transformation d'un fichier .desc dynamique en fichier statique

4.2.3 Intégration pour FIO

Le schéma d'intégration pour la stratégie FIO est très simple car le couplage sera résolu à l'extérieur d'iSIGHT à l'aide de l'enveloppe global décrit à la section 4.1.9. Bref, on donnera à iSIGHT uniquement un programme en Perl et un fichier global servant à la fois pour les entrées et pour les sorties (TDF).

La figure 4.13 illustre le schéma d'intégration de la stratégie FIO dans iSIGHT. Il n'y a qu'un seul « task » qui prend le même fichier en entrée et en sortie. On remarque que cette forme est identique à celle illustrée à la figure 4.2 et ceci est normal car une seule enveloppe est utilisé par iSIGHT.

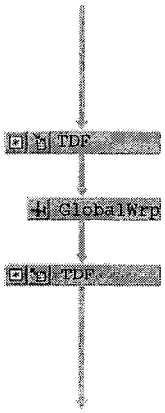

Figure 4.13 Schéma d'intégration de la stratégie FIO dans iSIGHT

4.2.4 Intégration pour DAO

Le problème est ici plus complexe à intégrer dans iSIGHT. Ceci vient principalement du fait qu'il y a maintenant trois programmes distincts. Donc, il faudra créer trois « SubTasks » et les faire communiquer avec le « Task » principal. Il y a également plus de variables et de contraintes.

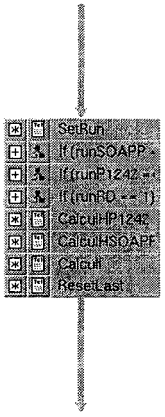

Figure 4.14 Schéma d'intégration de la stratégie DAO dans iSIGHT

La figure 4.14 montre le schéma d'intégration de la stratégie DAO dans iSIGHT. Il a trois « tasks » : un pour SOAPP, un pour P1242 et un pour RD. Les blocs de calcul « SetRun » et « ResetLast » servent à déterminer si l'on doit exécuter SOAPP et/ou P1242 et/ou RD pour l'itération en cours. Les autres blocs de calcul servent à calculer les contraintes de compatibilité $(t_i - r_i)$.

L'enveloppe globale utilisée pour FIO est exécutée à la fin de l'optimisation afin de résoudre le couplage aussi sévèrement qu'avec FIO.

CHAPITRE 5

RÉSULTATS ET ANALYSE

Les chapitres précédents ont permis de définir un modèle mathématique simplifié d'une turbine à gaz et d'expliquer l'intégration de ce modèle dans le logiciel iSIGHT. Ce chapitre présente les résultats obtenus en exécutant iSIGHT pour les stratégies FIO et DAO. Cette présentation sera suivie d'une comparaison des performances des deux méthodes.

5.1 Résultats et analyse pour FIO

La figure 5.1 présente la géométrie de la conduite de gaz et des disques avant et après l'optimisation. La solution optimale correspond au moteur le plus court avec les disques les plus hauts. Comme on le voit bien, l'optimiseur a cherché à éloigner le plus possible la ligne moyenne de d'écoulement de l'axe de rotation afin d'augmenter l'efficacité des turbines et ainsi minimiser le *sfc*.

Le tableau 5.1 compare certaines données présentées au tableau 2.5 avec les résultats de l'optimisation. Le *sfc* a diminué de 5.26% ce qui est très appréciable. Les rendements des deux turbines ont augmenté comme on pouvait s'y attendre car ils sont directement liés au *sfc*. Le poids ne connaît qu'une légère augmentation de 0.38%. Ceci est un heureux hasard car ni l'objectif ni les contraintes ne forçaient explicitement l'optimiseur à minimiser ou à maintenir le poids. Étant donné que les turbines tournent plus vite et que les ailettes sont plus loin de l'axe de rotation pour la solution finale, on aurait pu s'attendre à une augmentation beaucoup plus substantielle du poids. Cette faible augmentation s'explique probablement par le fait que les structures rotatives initiales étaient trop volumineuses, donc trop lourdes.

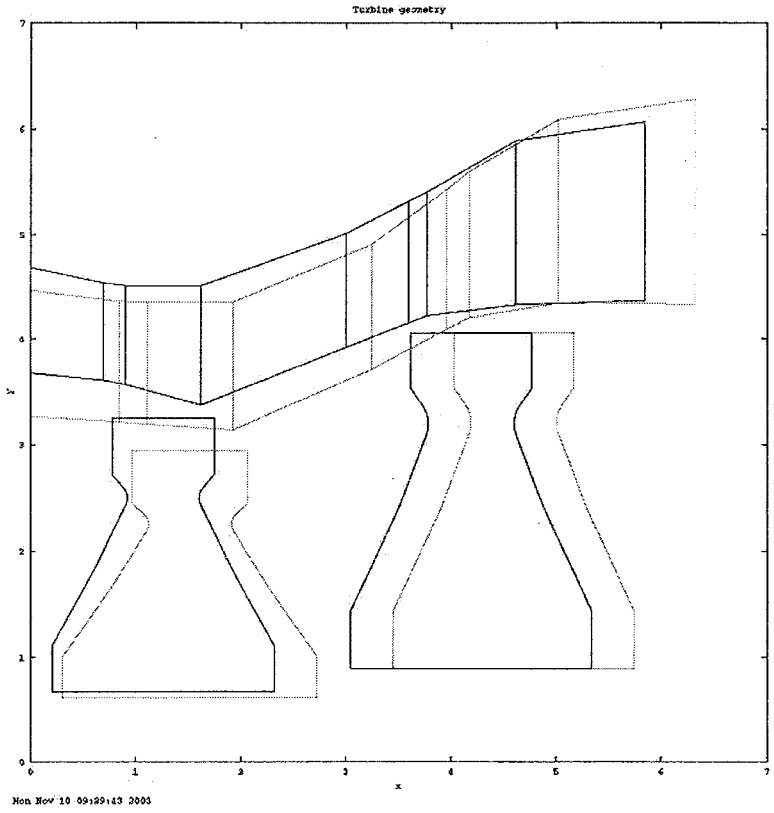

Figure 5.1 Solutions initiale et finale pour FIO

La figure 5.2 illustre la diminution relative du *sfc* en fonction du nombre d'itérations de l'optimiseur. Comme on le voit clairement, la solution est convergée. L'optimiseur s'est arrêté car la recherche de ligne dans la direction de descente trouvée requérait trop d'évaluation. Ceci signifie que le point trouvé n'est pas un minimum au sens mathématique du terme car le critère de convergence de Karush-Kuhn-Tucker (KKT) de premier ordre n'est pas rencontré. En effet, si tel était le cas, l'optimiseur se serait arrêté après le calcul des gradients sans faire une recherche de ligne supplémentaire.

Toutefois, tel qu'illustré à la figure 5.2, l'optimiseur a effectué beaucoup d'itérations sans amélioration notable du *sfc* et ceci indique que le minimum est fort probablement dans cette zone. En effet, puisque la précision exigée est de 0.1% et

Tableau 5.1 Solution finale vs initiale pour FIO

Variables	Description	$\frac{Finales-Initiales}{Initiales}$ (%)
sfc	Consommation spécifique de carburant	-5.26
HPT η	Rendement de la turbine haute pression	1.71
PT η	Rendement de la turbine de puissance	0.64
Poids	Poids total de tous les rotors	0.38

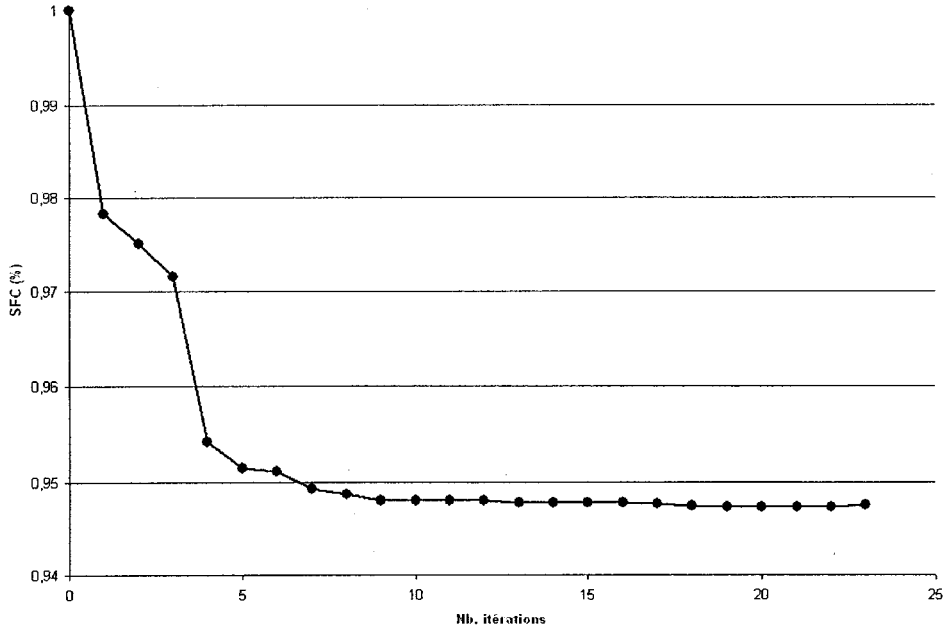

Figure 5.2 Variation du *sfc* durant l'optimisation pour la stratégie FIO

que la variation du *sfc* pour les dernières itérations est largement inférieure à cette valeur, on peut affirmer sans trop de risque qu'en tenant compte de la précision, on a atteint le minimum dans la zone. Le critère de KKT n'est probablement pas satisfait à cause du bruit numérique qui cause une imprécision sur le calcul des gradients.

Le tableau 5.2 montre le nombre d'évaluations effectuées par FIO ainsi que les différents programmes. La colonne FIO correspond au nombre de fois que NLPQL a appelé l'enveloppe globale via iSIGHT. La ligne « eval. opti. » correspond au nombre d'évaluations de NLPQL dans la direction de descente trouvée et représente donc l'ensemble des évaluations qui ne sont pas relatives au calcul des gradients. On remarque

que les 816 appels à l'enveloppe globale ont nécessité 2553 appels aux enveloppes de SOAPP et de P1242, ce qui signifie que le couplage entre les deux est résolu en 3.1 itérations en moyenne.

Il important de noter que l'enveloppe de P0831 n'a pas été utilisé. Ceci signifie que l'enveloppe de RD n'a pas jugé nécessaire de faire des analyses critiques en 2-D. On tire pour conclusion de ceci que la paramétrisation des disques est suffisamment bonne pour s'affranchir d'une analyse 2-D pour le cas étudié dans ce projet. Ceci n'est toutefois pas toujours vrai car il s'est avéré, lors d'essais préliminaire, que l'enveloppe de P0831 soit utilisé. Il faut donc assumer que la manière dont le problème FIO est formulé pour le cas présent joue également un rôle en permettant à l'optimiseur de se diriger dans une zone de l'espace de design ou la paramétrisation du disque suffit à éviter l'exécution de P0831.

Tableau 5.2 Nombre d'évaluations pour FIO

	FIO	SOAPP	P1242	RD	P0831
eval. grad.	748	2333	2333	748	0
eval. opti.	68	220	220	68	0
tot. eval.	816	2553	2553	816	0

Les temps de calcul sont présentés au tableau 5.3. La seule enveloppe nécessitant plus de 2 secondes de temps d'exécution est P0831 et il n'a pas été utilisé. Malgré cela, le temps moyen d'exécution d'une enveloppe est de 3.77 secondes. Ceci fait ressortir une certaine lenteur de l'intégration par iSIGHT au niveau des opérations de lecture et d'écriture et de traitement interne des données. On remarque par ailleurs, tout comme c'est le cas au tableau 5.2, que le temps d'optimisation est dominé par les efforts passés à calculer les gradients.

Tableau 5.3 Temps d'optimisation pour FIO

	temps (s)
gradient	23207
optimisation	2169
total	25376
temps/eval.	3.77

5.2 Résultats et analyse pour DAO

La figure 5.3 présente la géométrie de la conduite de gaz et des disques avant et après optimisation pour la stratégie DAO. La solution optimale correspond au moteur le plus court avec les disques les plus hauts. Comme on le voit bien, l'optimiseur s'est globalement comporté de la même manière que pour FIO, même s'il est évident à l'oeil nu que les deux solutions sont différentes.

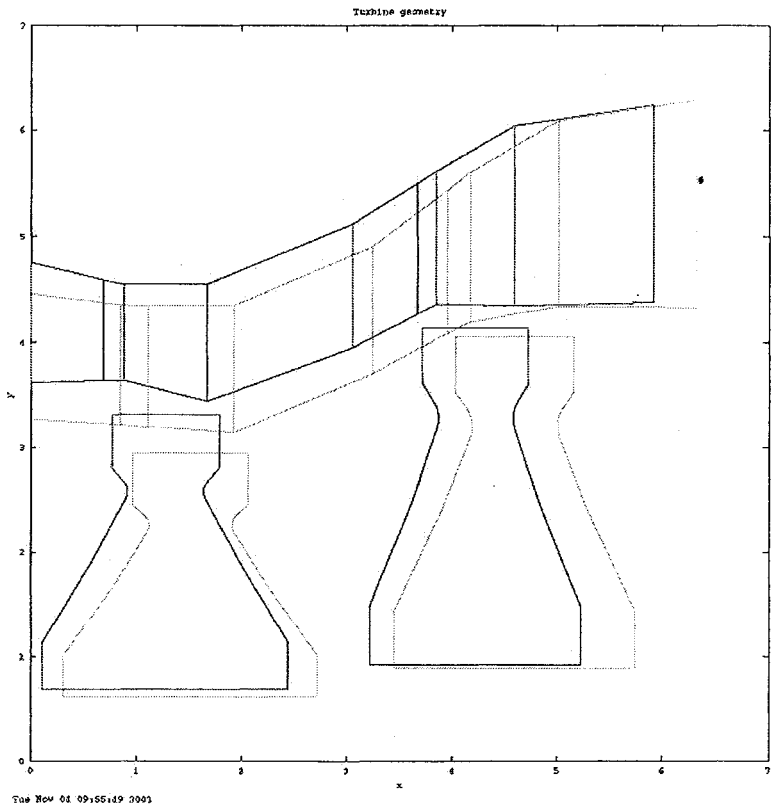

Figure 5.3 Solutions initiale et finale pour DAO

Le tableau 5.4 compare les données présentées au tableau 2.5 avec les résultats de l'optimisation avec DAO. Le *sfc* est réduit de 4.28%. Les rendements des deux turbines ont augmentés respectivement de 1.41 et 0.83%. Le poids connaît une légère augmentation de 0.47%, sans doute pour les mêmes raisons que celles évoquées pour la stratégie FIO.

Tableau 5.4 Solution finale vs initiale pour DAO

Variables	Description	$\frac{Finales-Initiales}{Initiales}$ (%)
sfc	Consommation spécifique de carburant	-4.28
HPT η	Rendement de la turbine haute pression	1.41
PT η	Rendement de la turbine de puissance	0.83
Poids	Poids total de tous les rotors	0.47

L'optimiseur s'est arrêté pour la même raison qu'avec FIO ; c'est-à-dire que la recherche de ligne dans la direction de descente trouvée requérait trop d'évaluations. On assume tout de même avoir atteint un minimum pour les mêmes raisons qu'avec FIO. En effet, on voit clairement à la figure 5.4 que le *sfc* a varié de moins de 0.1% durant les dernières itérations, ce qui suggère qu'aucune amélioration supplémentaire de l'objectif n'est possible dans cette zone de l'espace de design si l'on tient compte de la précision des programmes d'analyses.

Pour DAO, il est important de vérifier si toutes les contraintes de compatibilité sont respectées à l'optimum. La figure 5.5 montre la variation de la valeur mise à l'échelle des contraintes de compatibilité en fonction du nombre d'itérations de l'optimiseur ($H(adim.) = \frac{t_{final}-r_{final}}{r_{final}}$). Comme on le voit, les contraintes de compatibilité de 10 des 32 variables ne respectent pas la précision cible du design préliminaire qui est de 0.1%. On peut attribuer ce fait à l'imprécision des gradients pour ces variables pour le pas choisi, tel que montré pour certaines de ces variables à la figure III.1. Il est difficile et même voire impossible pour l'optimiseur de bien satisfaire les contraintes de compatibilité liées à ces variables dans de pareilles conditions. Heureusement, seules

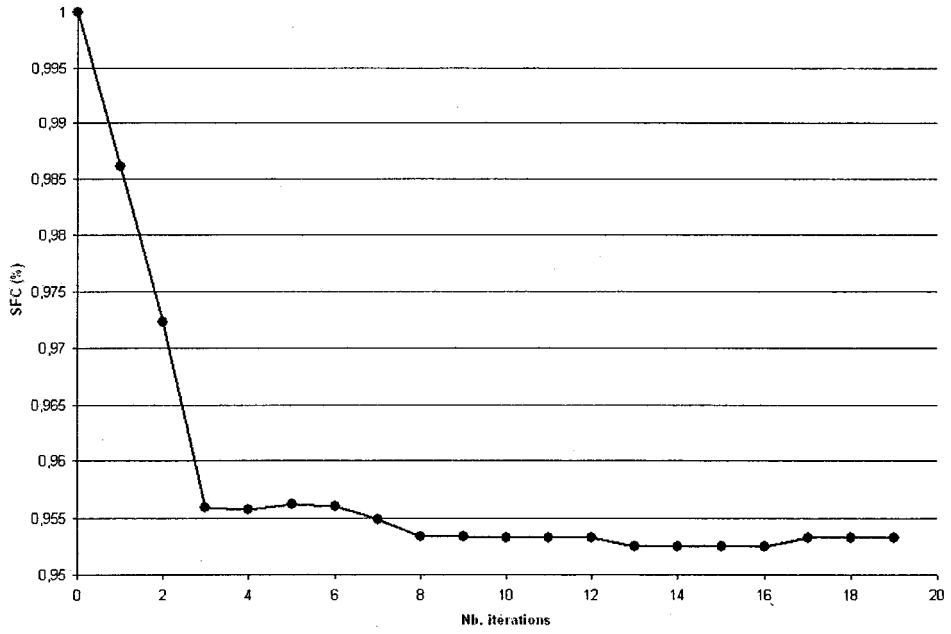

Figure 5.4 Variation du *sfc* durant l'optimisation pour la stratégie DAO

les contraintes de compatibilité sur Dp_1 et Dp_2 (rappelons que ces variables représentent la perte de charge pour chaque turbine) violent de beaucoup la borne de 0.1% et l'influence de ces variables est faible sur le *sfc*. On effectue tout de même une exécution de l'enveloppe globale utilisée pour FIO à la fin de l'optimisation avec DAO afin de s'assurer que toutes les contraintes de compatibilité sont satisfaites et que la valeur du *sfc* est bonne. En effet, il semble que le couplage est résolu avec plus de précision avec une analyse MDA couplée. Cette affirmation s'appuie sur deux tests dont les résultats sont illustrés aux figures 5.7 et 5.8. Sur la figure 5.7, on illustre le résultat d'une optimisation avec la stratégie FIO en partant de la solution obtenue avec la stratégie DAO. Similairementent, le résultat d'une optimisation avec la stratégie DAO en partant de la solution obtenue avec la stratégie FIO est reproduit sur la figure 5.8. Le résultat de cet exercice est que FIO n'a pas réussi à améliorer DAO mais que DAO à réussi à améliorer la solution FIO, qui était pourtant convergée. Le couplage est donc probablement résolu moins sévèrement avec DAO et ceci lui permet d'améliorer une solution FIO convergée. FIO n'arriva pas à améliorer DAO pour la

même raison. La figure 5.9 confirme le tout en montrant que DAO relâche quelque peu les contraintes de compatibilité par rapport à FIO.

En complément, la figure 5.6 montre les contraintes de compatibilité uniquement pour le *sfc* et les rendements des turbines. L'importance de ces variables pour le processus d'optimisation justifie une discussion additionnelle. En effet, pour que l'optimisation fonctionne bien, il faut que la contrainte de compatibilité sur le *sfc* soit satisfaite avec beaucoup de précision car le *sfc* est l'objectif. De plus, $t_{SOAPP_{sfc}}$ n'est une entrée d'aucun programme et donc n'influence aucune autre contrainte. Il faut donc que la contrainte sur le *sfc* soit bien satisfaite si on veut obtenir un résultat précis à la fin. La figure 5.6 illustre que cette contrainte est très bien statisfaite. Les rendements des turbines sont les variables couplées qui influencent le plus le *sfc* et de ce fait, ils sont donc très importants. Si les contraintes de compatibilité ne sont pas bien satisfaites, l'optimiseur aura tendance à diminuer le *sfc* de façon irréaliste en surestimant les valeurs des rendements ($t_{P1242_{\eta_{[1,2]}}}$). La figure 5.6 illustre qu'il est très difficile pour NLPQL de respecter ces contraintes avec une précision de 0.1%. La précision atteinte est plutôt de 0.3%, ce qui semble être suffisant pour bien prédire le *sfc*.

Le tableau 5.5 montre le nombre d'évaluations effectuées par DAO ainsi que les différents programmes. La colonne DAO correspond au nombre de fois que NLPQL a appelé une ou plusieurs enveloppes via iSIGHT. La ligne « eval. opti. » correspond au nombre d'évaluations de NLPQL dans la direction de descente trouvée et représente donc l'ensemble des évaluations qui ne sont pas attribuables au calcul des gradients. On remarque que les 1178 appels à des enveloppes par DAO ont nécessité 719 appels à l'enveloppe de SOAPP, 736 à celui de P1242 et 693 pour l'enveloppe de RD. Ceci montre bien que DAO appelle les différentes enveloppes uniquement lorsqu'une de leurs entrées a changée.

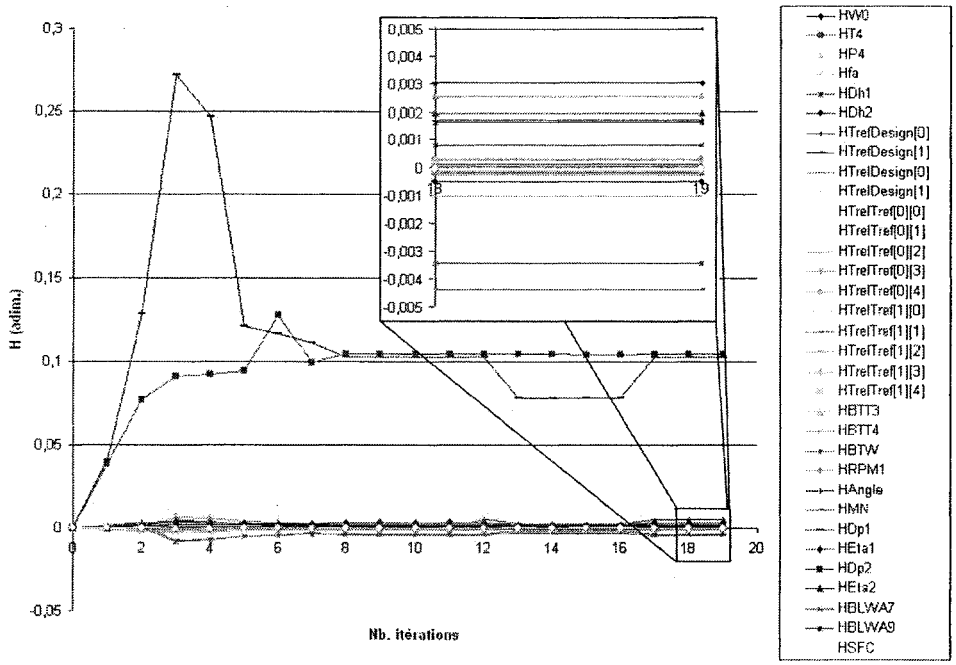

Figure 5.5 Variation des contraintes de compatibilité durant l'optimisation pour DAO

Il important de noter que l'enveloppe de P0831 n'a jamais été utilisé et ceci signifie que l'enveloppe de RD n'a pas jugé nécessaire de faire des analyses critiques en 2-D, tout comme pour FIO.

Tableau 5.5 Nombre d'évaluations pour DAO

	DAO	SOAPP	P1242	RD	P0831
eval. grad.	1105	646	663	748	0
eval. opti.	73	73	73	73	0
tot. eval.	1178	719	736	821	0

Les temps de calcul sont présentés au tableau 5.6. Rappelons que seule l'enveloppe de P0831 s'exécute en plus de 2 secondes et qu'il n'a pas été utilisé. Malgré cela, le temps moyen d'exécution d'un programme est de 21.88 secondes. Ce problème de lenteur a également été observé pour FIO et est attribuable en grande partie à la lenteur des opérations de lecture et d'écriture effectuées par iSIGHT. Par contre, le problème

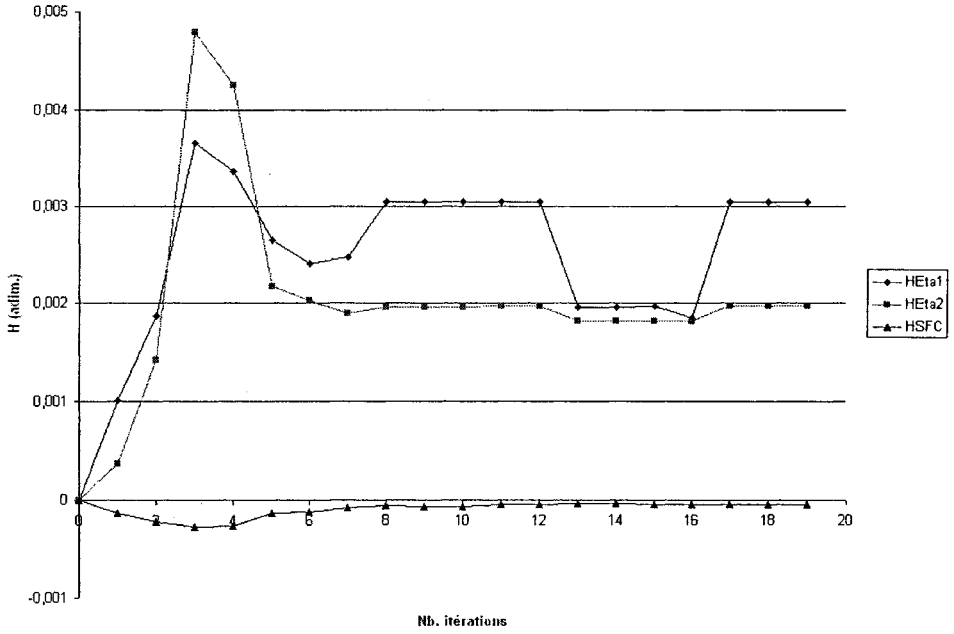

Figure 5.6 Variation des contraintes de compatibilité du *sfc* et des rendements $\eta_{[1,2]}$ durant l'optimisation pour DAO

est plus important pour la stratégie DAO. En effet, la différence principale entre les intégrations pour FIO et DAO est que pour FIO, iSIGHT n'a à lire et à écrire qu'une seule fois dans un fichier (le TDF) alors que le reste des opérations de lecture et d'écriture est géré par l'enveloppe global de FIO. Au contraire pour DAO, iSIGHT appelle directement les différentes enveloppes et effectue toutes les opérations de lecture et d'écriture. Ceci ralentit de beaucoup l'optimisation, le temps moyen d'exécution d'un programme passant de moins de 2 secondes hors de iSIGHT à près de 22 secondes via iSIGHT.

Tableau 5.6 Temps d'optimisation pour DAO

	temps (s)
gradient	44702
optimisation	5119
total	49821
temps/eval.	21.88

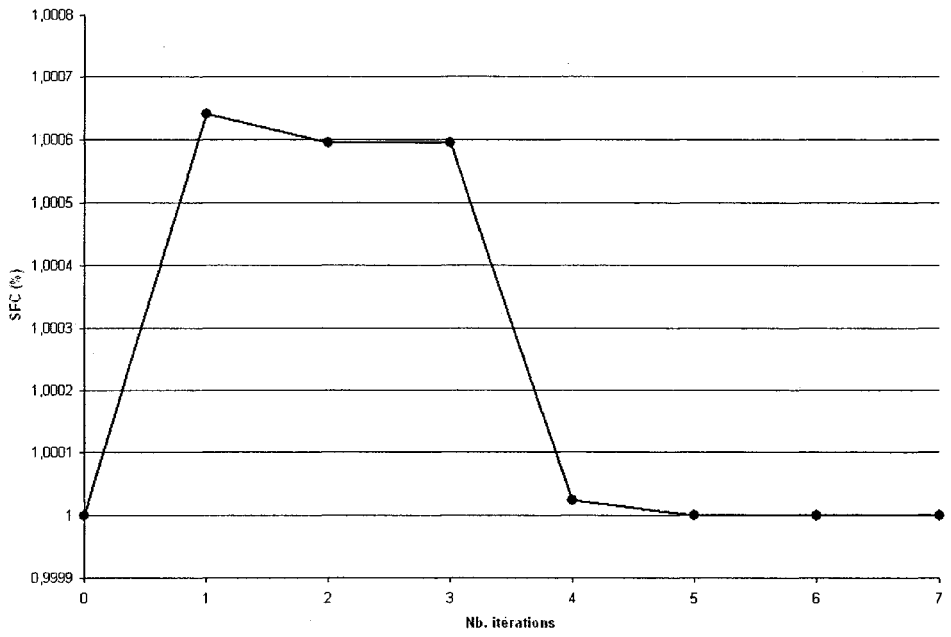

Figure 5.7 Variation du *sfc* durant l'optimisation avec FIO en partant de la solution avec DAO

5.3 Comparaison de FIO et de DAO

La figure 5.10 illustre la section de la turbine obtenue via les deux formulations. La figure 5.11 montre l'évolution de la valeur mise à l'échelle du *sfc* pour FIO et DAO. Comme on le voit, les deux formulations n'ont pas convergé vers le même minimum et le *sfc* minimum trouvé par FIO est meilleur celui trouvé par DAO. Les formulations FIO et DAO étant différentes, elles n'ont pas parcouru le même chemin dans l'espace de design et n'ont pas convergé vers le même minimum, comme on le voit clairement à la figure 5.10. Il est donc difficile de comparer les deux stratégies uniquement en se basant sur le *sfc* à l'optimum.

Afin de comparer de façon plus équitable les deux stratégies, deux relations permettant d'approximer le nombre d'appels aux enveloppes effectués par FIO et DAO ont été construites. Ces approximations sont définies à l'équation 5.1. Avec ces équations, on pourra observer les différents paramètres qui influencent le nombre d'appels

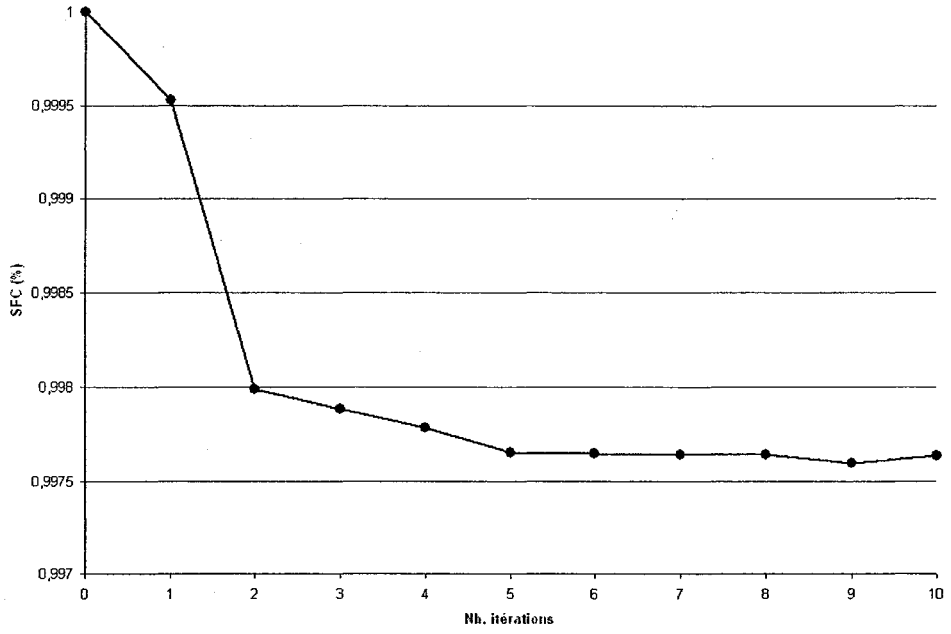

Figure 5.8 Variation du *sfc* durant l'optimisation avec DAO en partant de la solution avec FIO

aux enveloppes. Il sera ainsi plus facile de comparer FIO et DAO en terme de performance.

$$\begin{aligned} Nb_{appels_{FIO}} &= (2X+1)(N_{FIO}M_{FIO} + P_{FIO} + Q_{FIO}) \\ Nb_{appels_{DAO}} &= N_{DAO}(M_{DAO} + Q_{DAO}) + 3P_{DAO} \end{aligned} \tag{5.1}$$

où

X = nombre moyen d'itérations pour résoudre le couplage

N_{FIO} = le nombre de variables pour FIO

$N_{DAO} = 2s + t + l + 1$ où s = nombre de variables de type s, l = nombre de variables de type l, t = nombre de variables de type t

$M_{[FIO,DAO]}$ = nombre d'itérations d'optimisation pour FIO ou DAO

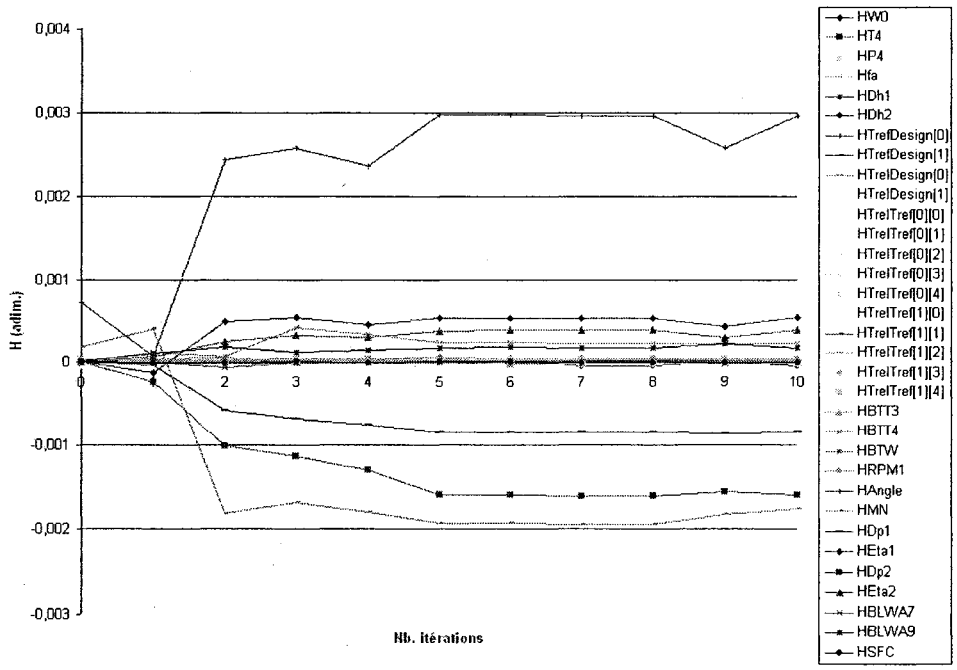

Figure 5.9 Variation des contraintes de compatibilité durant l'optimisation pour DAO en partant de la solution avec FIO

$P_{[FIO,DAO]}$ = nombre de recherches de ligne pour FIO ou DAO

$Q_{[FIO,DAO]}$ = nombre d'étapes de NLPQL utilisées pour FIO ou DAO

On voit clairement à l'équation 5.1 que le nombre d'appels dépend de plusieurs paramètres. Pour le problème étudié dans cette étude, ces paramètres prennent les valeurs données au tableau 5.7 où les valeurs entre parenthèses correspondent aux résultats réels obtenus.

Tableau 5.7 Valeurs des paramètres de l'équation 5.1 pour FIO et DAO

FIO			DAO		
X	=	3.1			
N_{FIO}	=	34	N_{DAO}	=	92
M_{FIO}	=	20	M_{DAO}	=	19
P_{FIO}	=	68	P_{DAO}	=	73
Q_{FIO}	=	2	Q_{DAO}	=	3
$Nb_{appels_{FIO}}$	=	5875 (5922)	$Nb_{appels_{DAO}}$	=	2243 (2276)

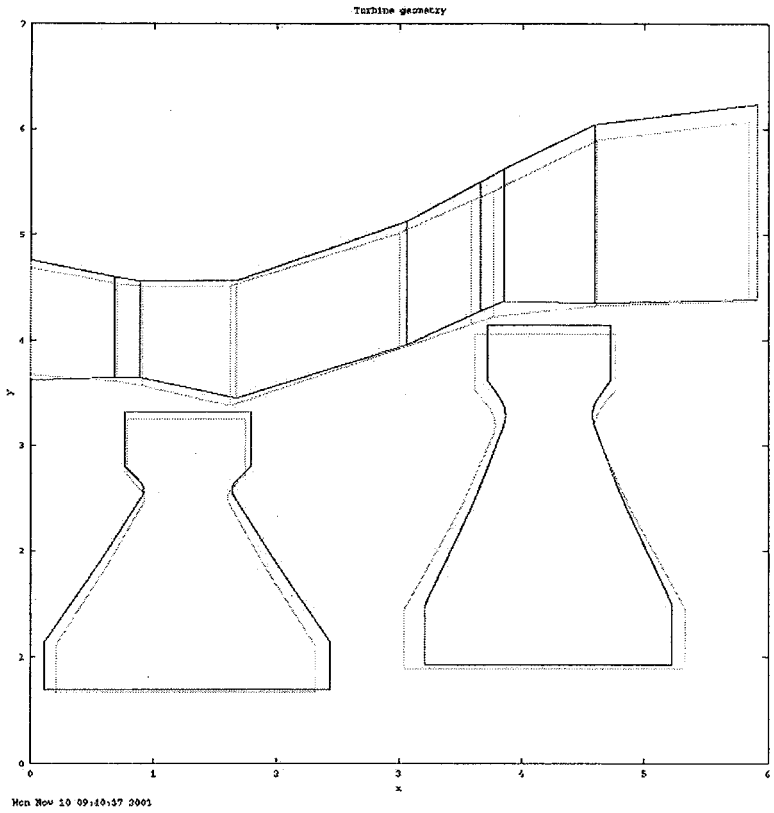

Figure 5.10 Comparaison des solutions obtenues avec FIO et DAO (DAO est en trait épais)

On voit donc au tableau 5.7 que la stratégie DAO est plus efficace que la stratégie FIO au niveau du nombre d'évaluations pour le problème étudié. En fait, FIO doit faire, en moyenne, environ 2.5 fois plus d'appels aux enveloppes que DAO à chaque itération. Le seul facteur qui n'est pas explicitement pris en compte dans les relations définies à l'équation 5.1 est le nombre de contraintes et la difficulté de l'optimiseur à les satisfaire. Dans les faits, $M_{[FIO,DAO]}$ et $P_{[FIO,DAO]}$ sont des fonctions des facteurs liées aux contraintes car elles dépendent de la capacité de NLPQL à résoudre le problème. Or, pour le présent problème, il semble que le fait qu'il y ait beaucoup plus de contraintes pour DAO ne cause pas de problème à NLPQL. On peut émettre l'hypothèse que ceci est dû au fait que les contraintes additionnelles de DAO sont celles de compatibilité et que celles-ci sont respectées de façon plus sévère avec FIO qu'avec DAO (voir la section 5.2). On tire de ces remarques qu'il aurait fallu que les

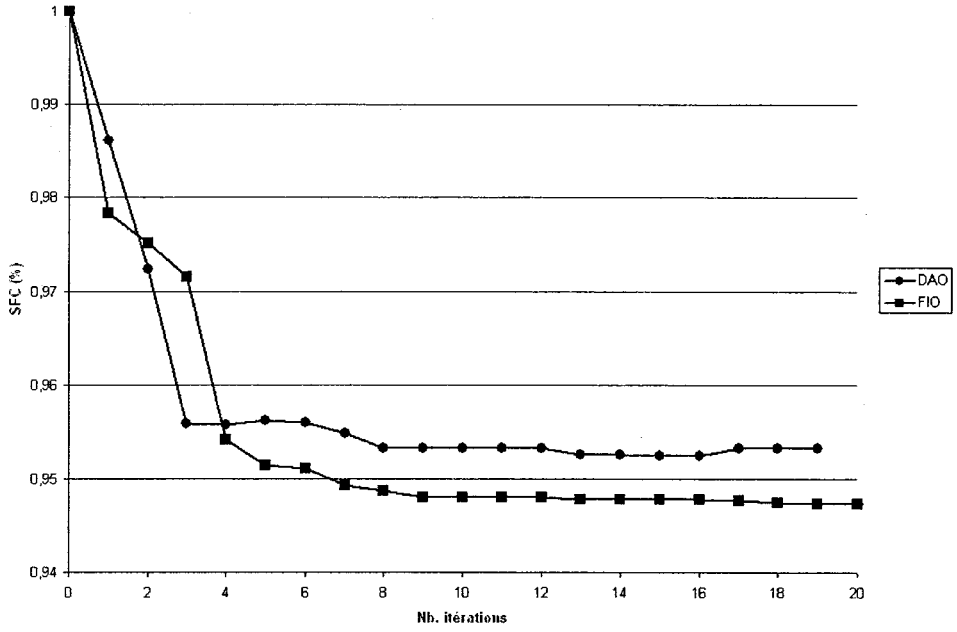

Figure 5.11 Comparaison de la variation du *sfc* durant l'optimisation pour FIO et DAO

contraintes de compatibilité compliquent beaucoup plus la tâche de NLPQL pour que FIO domine DAO au chapitre du nombre d'appels aux enveloppes.

En regardant la figure 5.11, on remarque qu'il n'y a pas d'amélioration significative du *sfc* après les 10 premières itérations de NLPQL. On prend donc les 10 premières itérations comme base de comparaison entre FIO et DAO pour le reste de l'analyse des résultats. Le tableau 5.8 donne les valeurs des paramètres de l'équation 5.1 pour FIO et DAO pour les 10 premières itérations. On voit clairement dans ce tableau que FIO effectue environ 2.5 fois plus d'appels aux programmes d'analyse que DAO.

On peut aussi comparer les temps de calculs pour FIO et DAO. Pour ce faire, il faut cependant éliminer le temps requis par iSIGHT pour lire et écrire, sans quoi la comparaison ne sera pas représentative des performances réelles de FIO et DAO. En effet, comme il a été observé lors de la présentation des résultats pour les deux stra-

Tableau 5.8 Valeurs des paramètres de l'équation 5.1 pour FIO et DAO après 10 itérations

FIO			DAO		
X	=	3.1			
N_{FIO}	=	34	N_{DAO}	=	92
M_{FIO}	=	10	M_{DAO}	=	10
P_{FIO}	=	19	P_{DAO}	=	26
Q_{FIO}	=	1	Q_{DAO}	=	1
$Nb_{appels_{FIO}}$	=	2592	$Nb_{appels_{DAO}}$	=	1090

tégies, les opérations de lecture et d'écriture dans le logiciel iSIGHT ralentissent par un facteur d'environ 2 l'optimisation de FIO et par un facteur d'approximativement 11 pour DAO. Il sera toutefois possible, dans les projets futurs, de contourner cette limitation de iSIGHT en lui fournissant directement un code de « parsing » en Perl au lieu d'en définir un en langage Mdol. On pourra ainsi probablement éliminer en totalité le problème de lenteur des opérations de lecture et d'écriture dans iSIGHT. Ceci permettra d'obtenir un temps moyen d'exécution des enveloppes des programmes d'analyse inférieur à 2 secondes. On peut donc estimer le temps approximatif qu'aurait dû prendre les deux optimisations en réalité simplement en multipliant les valeurs $Nb_{appels_{FIO}}$ et $Nb_{appels_{DAO}}$ données au tableau 5.8 par 2 secondes. On obtient ainsi que DAO est environ 2.5 fois plus rapide que FIO pour le problème étudié.

On peut conclure jusqu'ici que la stratégie DAO est largement supérieure à FIO en terme de nombre d'appels aux programmes d'analyse et de temps de calcul. Par contre, il ne faut pas perdre de vue que la solution obtenue avec la stratégie FIO est supérieure à celle obtenue avec la stratégie DAO par plus de 1%, ce qui est non négligeable. En fait, on a montré que si on donne comme points de départ à FIO le point minimum trouvé par DAO, alors FIO ne bouge pas. Ceci démontre que DAO a bien trouvé un minimum valable, différent de celui trouvé par FIO. Il faut toutefois noter que c'est potentiellement le fruit du hasard qui fait en sorte que FIO descend vers un meilleur minimum que DAO. En effet, tel qu'illustré à la figure 5.12, la dis-

tance à parcourir ainsi que les obstacles sur le parcours afin d'atteindre un minimum dépendent du point de départ dans l'espace de design. Or, les formulations FIO et DAO sont différentes et donc n'empruntent pas nécessairement le même chemin. Ceci avantage potentiellement une stratégie par rapport à l'autre selon l'endroit où se situe le point de départ en plus de donner des minimums différents.

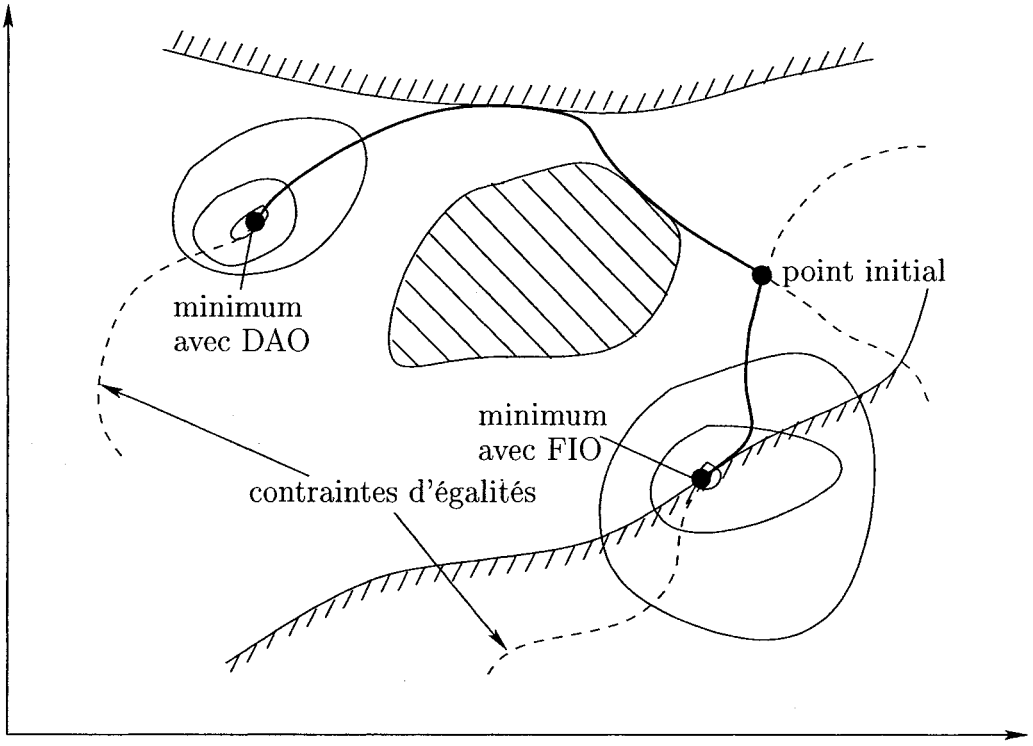

Figure 5.12 Exemple de chemins ayant été suivies par FIO et DAO dans l'espace de design

Il reste un point à observer afin de comparer les performances des formulations FIO et DAO et c'est le potentiel de parallélisation. En effet, dans les deux cas il est possible d'effectuer le calcul des dérivées de façon parallèle. Pour FIO, le nombre de dérivées à calculer à chaque itération, donc d'appels à l'enveloppe globale, est 34 car il y a 34 variables. Pour DAO, ce nombre est de 92 car les variables partagées (de type s) nécessitent l'exécution de 2 enveloppes et une de ces variables nécessite même un appel aux trois enveloppes ($RPM_{[2]}$). Il est donc possible de diviser par

un facteur d'environ 34 le temps requis pour faire une analyse FIO et par 92 pour DAO. Dans ces conditions, DAO devient environ 6.5 fois plus rapide que FIO, tel que montré au tableau 5.9 pour la situation où on regarde le temps requis par FIO et DAO pour effectuer 10 itérations. On se retrouve ainsi avec des temps d'analyse de l'ordre de quelques minutes. On peut faire le même raisonnement pour le temps total des deux optimisations. On obtiendrait ainsi une solution, convergée par NLPQL, en 348 secondes pour FIO et en 49 secondes pour DAO, en temps corrigés toujours. Il est important de noter que ces temps corrigés peuvent encore être améliorés en parallélisant l'enveloppe de RD, lui permettant ainsi de faire le design des deux étages de turbine simultanément. Il est donc potentiellement possible d'obtenir une solution convergée pour DAO en moins d'une minute.

Tableau 5.9 Temps d'optimisation corrigés pour FIO et DAO parallélisés

	FIO (s)	DAO (s)
temps total	348	49
temps pour 10 itérations	154	24

La conclusion finale de ce chapitre va comme suit :

- Pour le problème traité, le minimum trouvé par la formulation FIO est meilleur que celui trouvé par DAO en terme de réduction du *sfc*. Les deux formulations n'ont pas convergé vers le même minimum. Ceci démontre la non-convexité du problème et suggère à l'avenir d'en tenir compte dans les essais.
- DAO a l'avantage sur FIO en terme de nombre d'appels aux enveloppes et de temps de calcul corrigés pour une solution convergée par NLPQL.
- Si on tient compte du potentiel de parallélisation des deux méthodes, on observe que FIO mettra en général 6.5 fois plus de temps (en temps corrigés) que DAO pour un problème similaire à celui traité dans ce projet. DAO a donc le potentiel de fournir une solution acceptable beaucoup plus rapidement que FIO.

CONCLUSION

Le but premier de cette maîtrise était d'explorer et de comparer les performances des stratégies DAO et FIO pour un problème multidisciplinaire de grande taille avec un couplage léger. Ceci permet d'explorer le comportement de FIO et DAO sur une classe de problèmes différente de celles rencontrées jusqu'à présent dans la littérature. De plus, les essais se sont déroulés à l'aide de modules d'analyse industriels, ce qui pouvait aussi apporter des difficultés supplémentaires en raison du bruit dans la réponse de tels programmes.

Pour ce faire, l'optimisation d'une turbine à gaz au stade de design préliminaire a été effectuée chez P&WC. L'enrobage des programmes d'analyses SOAPP, P1242, RotorDesigner et P0831 a été effectué en langage Perl, définissant ainsi une enveloppe pour chacun d'eux. Ces derniers ont ensuite été facilement intégrés dans le logiciel iSIGHT pour les formulations FIO et DAO et un moteur fictif. Une mise à l'échelle des variables et des contraintes ainsi qu'une réduction du nombre de variables pour FIO et DAO ont permis de faciliter l'optimisation avec NLPQL. Plusieurs ajustements sur le poids des différentes contraintes ont ensuite été nécessaires afin de converger rapidement vers un minimum local.

Du côté de l'intégration, le présent travail a permis de mettre en évidence la lenteur des opérations de lecture et d'écriture dans iSIGHT. La conclusion à en tirer est qu'il est souhaitable de minimiser le nombre d'opérations de lecture et d'écriture réalisées par iSIGHT pour obtenir une optimisation plus rapide. Le logiciel iSIGHT n'en demeure pas moins un puissant outil d'optimisation et d'analyse, permettant d'expérimenter rapidement une multitude de technique d'optimisations et d'exploration du design. De plus, cette conclusion s'applique principalement lorsque les programmes d'analyse sont rapides, de l'ordre de quelques secondes. Si les programmes deman-

daient plusieurs minutes d'exécution, alors le délai causé par la lenteur des opérations de lecture et d'écriture par iSIGHT deviendrait négligeable. Pour le cas similaire à celui présenté ici, avec des programmes rapides, il serait alors important de confier au langage Perl l'ensemble des tâches de « parsing » et d'éviter d'utiliser iSIGHT pour accélérer l'optimisation en terme de temps total.

Du point de vue de l'optimisation, l'algorithme NLPQL s'est avéré efficace en réduisant par environ 4 à 5% le *sfc* en quelques itérations pour FIO et DAO. Les résultats ont démontrés que DAO fournit une solution en environ 2.5 fois moins d'appels aux enveloppes que FIO pour la classe de problèmes étudiés. La stratégie FIO a toutefois battu la stratégie DAO par plus de 1% au niveau du *sfc* pour le seul cas étudié. Ceci ne doit pas nécessairement permettre de conclure à la supériorité de FIO puisque les deux techniques n'ont pas convergé vers le même minimum et qu'un seul point de départ a été utilisé. Un essai a été réalisé en démarrant de la solution DAO et en appliquant l'algorithme FIO et ce dernier n'a pas amélioré la solution trouvée par DAO, montrant clairement qu'il s'agit de deux optimums locaux distincts. C'est donc potentiellement le fruit du hasard qui a avantagé FIO par rapport à DAO au niveau de la minimisation du *sfc* pour le point de départ choisi.

Dans ce genre de problème, il serait donc pertinent de développer une méthode pour générer un ensemble de points de départs prometteurs dans l'espace de design. Il faut cependant noter que le point de départ utilisé dans cette étude était réalisable et que des tests en partant de points irréalisables sur le plan des structures rotatives se sont tous soldés par un échec de NLPQL à trouver un point faisable. Il est donc nécessaire d'étudier plus en profondeur les causes de ces difficultés afin de trouver une solution. Les avenues de recherche à ce sujet devront sans doute passer par l'étude de méthodes d'exploration de l'espace de design, tels les méthodes génétiques et les algorithmes DOE.

Durant ce projet, les tests ont porté principalement sur un problème réduit mais des tests préliminaires ont également été effectués pour le problème avec toutes les variables. Ces tests n'ont pas donnée d'aussi bons résultats que pour le problème réduit et il serait donc pertinent dans l'avenir d'explorer les causes de ce phénomène. Notamment, il faudrait étudier plus en détail l'optimisation des attaches des ailettes qui n'a pas vraiment été poussée.

Il a fallu beaucoup de réglages aux formulations FIO et DAO avant que NLPQL obtienne rapidement un bon minimum. Ces ajustements ont toutefois été testés uniquement sur le moteur étudié dans ce projet et on ne sait donc pas s'ils sont valides pour tous les moteurs. Il serait donc pertinent de tester les formulations FIO et DAO sur d'autres moteurs et ainsi trouver des réglages s'adaptant à une grande variété de turbines.

Les difficultés rencontrées pour le problème complet et pour les ajustements de FIO et DAO sur le problème réduit peuvent indiquer que la taille du problème est un peu grande pour permettre à un optimiseur unique de varier adéquatement toutes les variables. Il serait intéressant de décomposer le problème par l'utilisation de méthodes multi-niveaux. On simplifierait ainsi la tâche des optimiseurs en subdivisant l'espace de design en morceaux de tailles réduites. L'étude de méthodes multi-niveaux est donc recommandée car ces méthodes risquent fort d'être mieux adaptées à des problèmes de plus grandes tailles que celui étudié ici, comme prévoient le faire les ingénieurs de P&WC [1] en incluant l'analyse de la « fan », des compresseurs axial et radial, du poids, du coût, du « nozzle », etc.

Finalement, il serait intéressant, et très pertinent, d'exploiter le potentiel de parallélisation de FIO et DAO dans des projets futurs de taille similaire ou plus grand

en parallélisant le calcul des dérivées. On pourrait ainsi accélérer par un facteur de 30 l'optimisation de FIO et par un facteur de 90 celle de DAO dans le cas d'un problème de taille similaire à celui traité et en supposant que suffisamment d'ordinateurs sont disponibles pour les calculs. Le point important est que le temps de calcul des gradients augmente avec le nombre de variables et que la parallélisation a le potentiel d'annuler cette augmentation et donc de rendre le temps de calcul des gradients presque indépendant du nombre de variables de design. Pour les problèmes similaires à celui traité ici, ceci permettrait théoriquement d'obtenir une solution convergée en moins d'une minute pour DAO en autant que le nombre d'ordinateurs est suffisant et que les opérations de lecture et d'écriture sont effectuées en Perl plutôt qu'en Mdol. Cette parallélisation permettrait aux ingénieurs de pouvoir vraiment utiliser la MDO dans un contexte de design préliminaire (PMDO) en obtenant des designs préliminaires optimaux et pleinement réalisables très rapidement.

RÉFÉRENCES

[1] PANCHENKO, Y. et al. (2002). Preliminary Multi-Disciplinary Optimization in Turbomachinery Design. *DETC 2002/ISD-34436.*

[2] ALEXANDROV, N. M. et LEWIS, R. M. (2000). Analytical and Computational Aspects of Collaborative Optimization. Rapport technique *NASA/TM-2000-210104*, NASA Langley Research Center.

[3] KORTE, J.J. , SALAS, A.O., DUNN, H.J. et ALEXANDROV, N. M. (1997). Multidisciplinary Approach to Linear Aerospike Nozzle Optimization. Dans *American Institute of Aeronautics and Astronautics Journal.*

[4] KORTE, J. J. , WESTON, R. P., et ZANG, T. A. (1997). Multidisiciplinary Optimization Methods for Preliminary Design. Dans *AGARD Interpanel Symposium "Future Aerospace Tecnology in the Service of the Alliance"*, École Polytechnique, Paris, France.

[5] SOBIESZCZANSKI-SOBIESKI, J. et HAFTKA, R. T. (1996). Multidisciplinary Aerospace Design and Optimization : Survey of Recent Developments. *96–0711*, Reno, NV.

[6] KROO, I., ALTUS, S., BRAUN, R., GAGE, P. et SOBIESKI, I. (1994). Multidisciplinary Optimization Methods for Aircraft Preliminary Design. Dans *American Institute of Aeronautics and Astronautics Journal,94–4325.*

[7] PADULA, S.L., KORTE, J. J., DUNN, H. J. et SALAS, A. O. (1999). Multidisciplinary Optimization Branch Experience Using iSIGHT Software. Rapport

technique *TM-1999-209714*, NASA.

[8] WALSH, J. L., YOUNG, K. C., PRITCHARD, J. I., ADELAMN, H. M. et MANTAY, W. R. (1995). Integrated Aerodynamic/Dynamic/Structural Optimization of Helicopter Rotor Blades Using Multilevel Decomposition. Rapport technique *3465*, NASA.

[9] TAYLA, S. S., CHATTOPADHYAY, A. et RAJADAS, J. N. (2002). Multidisciplinary Design Optimization Procedure for Improved Design of a Cooled Gas Turbine Blade. Dans Taylor & FRANCIS, éditeur, *Engineering Optimization*, volume *34(2)*.

[10] BRAUN, R. D. (1996). *Collaborative Optimization : An Architecture for Large-Scale Distributed Design.* Thèse de doctorat, Stanford University.

[11] TRIBES, C. (2002). Evaluation of Multidisciplinary Design Optimization MDO Decomposition Methodologies on Analytical Problems. Rapport technique, Centre de Recherche en Calcul Appliqué.

[12] ALEXANDROV, N. M. et LEWIS, R. M. (2000). Analytical and Computational Properties of Distributed Approaches to MDO. *2000-4718.*

[13] ALEXANDROV, N. M. et LEWIS, R. M. (2000). Algorithmic Perspective on Problem Formulations in MDO. *2000-4719.*

[14] ALEXANDROV, N. M. et LEWIS, R. M. (1999). Comparative Properties of Collaborative Optimization and Other Approaches to MDO. Dans *First ASMO*

UK/ISSMO Conference on Engineering Design Optimization. MCB University Press.

[15] CRAMER, E. J., DENNIS JR., J. E., FRANK, P. D., LEWIS, R. M. et SHUBIN, G. R. (1993). Problem formulation for multidisciplinary optimization. Dans *AIAA Symposium on Multidisciplinary Design Optimization.*

[16] RÖHL, P. J., HE, B. et FINNIGAN, P. M. (1998). A Collaborative Optimization Environment for Turbine Engine Development. *98–4734.*

[17] ALEXANDROV, N. M. et LEWIS, R. M. (2003). Dynamically Reconfigurable Approach to Multidisciplinary Problems. *2003-3431.*

[18] SPELLUCCI, P. (1999). *DONLP2 Short Users Guide.* Program and user's guide available as `donlp2.tar` at `ftp://plato.la.asu.edu/pub/donlp2/`.

[19] SCHITTKOWSKI, K. (2001). *NLPQLP : A New Fortran Implementation of a Sequential Quadratic Programming Algorithm for Parallel Computing.*

[20] KODIYALAM, S. (1998). Evaluation of Methods for Multidisciplinary Design Optimization (MDO), phase I. Rapport technique *NASA/CR-1998-208716*, NASA Langley Research Center.

[21] ALEXANDROV, N. M. et KODIYALAM, S. (1998). Initial Result of an MDO Method Evaluation Study. Dans *American Institute of Aeronautics and Astronautics Journal, AIAA-98-4884.*

[22] REED, J. A. Java Gas Turbine Simulator : Engine Component Mathematical Model.

[23] REED, J. A. Java Gas Turbine Simulator : Numerical Solvers.

[24] KIM, J. H., SONG, T. W., KIM, T. S. et RO, S. T. (2001). Model Development and Simulated of Transiant Behavior of Heavy Duty Gas Turbines. *Journal of Engineering for Gas Turbines and Power, 123(589).*

[25] EVEKER, K. M. et NETT, C. N. (1991). Active Surge Control/Rotating Stall Avoidance in Aircraft Gas Turbine Engines. Dans *American Control Conference.*

[26] KURZKE, J. Gasturb. Program and user's guide available at `http://www.gasturb.de`.

[27] DENTON, J.D. (1993). Loss Mechanisms in Turbomachines. Dans *International Gas Turbine and Aeroengine Congress and Exposition, 93-GT-435*, Cincinnati, Ohio.

[28] KACKER, S.C. et OKAPUU, U. (1982). A Mean Line Prediction Method for Axial Flow Turbine Efficiency. *Journal of Engineering for Power, 104(113).*

[29] PAPALAMBROS, P. Y. et WILDE, D. J. (2000). *Principles of Optimal Design : Modeling and Computation.* Cambridge University Press, 2^{nd} édition.

[30] TURCOTTE, J. (2003). Process Report 2. Rapport technique, Pratt & Whitney Canada.

[31] WALL, L., CHRISTIANSEN, T. et ORWANT, J. (2001). *Programming Perl*, chapitre 18. O'Reilly, third édition.

[32] CORMIER, P.-L. (2002). Pré-processeur facilitant l'utlisation de rotordesigner dans un processus d'optimisation préliminaire. Rapport technique, École Polytechnique de Montréal. Projet de fin d'étude.

ANNEXE I

PARAMÉTRISATION

Malgré que l'on ne soit qu'au stade préliminaire de conception, il y a un nombre élevé de variables apparaissent dans la paramétrisation. La présente annexe vise à les identifier toutes. Par contre, à ce stade du design, il n'est pas vraiment pertinent de faire varier l'ensemble de ces variables. C'est pourquoi, certaines d'entre elles seront exprimées comme une fonction des autres. On vient donc ici tenter de minimiser au maximum le nombre de variables. Il est important de noter que cette étape est en totalité basée sur l'expérience des ingénieurs, en occurrence de M. Michel Dion et M. Ghislain Plante pour ce projet.

I.1 SOAPP

Puisque SOAPP est un programme 0-D, il n'y a pas de paramétrisation géométrique à faire. La paramétrisation consiste plutôt à décider quelles variables doivent être considérées fixes par SOAPP. Normalement, lors d'un design préliminaire, une puissance est imposée. Cette puissance est fixée et sera donc un paramètre tout au long de l'optimisation. La puissance dépend de deux autres variables thermodynamiques : le débit d'air en entrée (W_C) et la température à la sortie de la chambre de combustion (T_4). pour ce projet, on a décidé de fixer le débit d'air et donc de confier à SOAPP le soin de trouver la température T_4 qui correspond. Ceci correspond à ce qui est fait normalement par les ingénieurs.

Il est important de noter que le fait de fixer W_C ne signifie pas que le débit devient un paramètre fixe pour l'optimiseur. Ceci signifie seulement que SOAPP doit considérer le débit W_C ainsi que la puissance du moteur comme fixe et chercher le T_4 correspondant en se servant du T_4 fourni en entrée comme point de départ. En

résumé, W_C est donc une variable qui est une entrée de SOAPP alors que T_4 devient une sortie de SOAPP.

I.2 Conduite de gaz « gaspath »

Il y a deux moyens de définir la conduite de gaz : par les rayons ou par les angles. Les aérodynamiciens préfèrent une paramétrisation par les angles car ils peuvent ainsi les fixer directement à la valeur désirée. Les ingénieurs de structure préfèrent de leur côté les rayons car en structure rotative tout tourne autour des rayons. Le choix était déjà fait au début de cette maîtrise[32] et ce sont les angles qui l'emportent.

Dans les faits, les deux paramétrisations sont équivalentes en terme du nombre de variables. Des différences s'observent au niveau du type de chaque variable (locale, couplée ou partagée) lors de la décomposition du problème. L'impact de ces différences n'a pas été étudié dans ce projet. D'autres différences peuvent survenir au niveau du type de contrainte en sortie à imposer (angle constant, rayon constant). Dans ce projet, ce type de contrainte n'est pas utilisé. On calcule les différentes dimensions de la conduite de gaz à l'aide des équations I.1, I.2 et I.3.

$$B_{x-gap_i} = \gamma B_{x-v_i}$$

$$AL_{5-6_i} = \begin{cases} \lambda \dfrac{2\pi}{NOV_{i+1}} \dfrac{R_{h-0_{i+1}} + R_{h-2_{i+1}} + R_{t-0_{i+1}} + R_{t-2_{i+1}}}{4} & \text{si } i > 0 \wedge AL_{5-6_i} = 0, \\ AL_{5-6_i} & \text{sinon.} \end{cases} \tag{I.1}$$

où

γ est un % de B_{x-v_i}

λ est un % de la distance, au rayon moyen, entre chaque aube de l'étage suivant

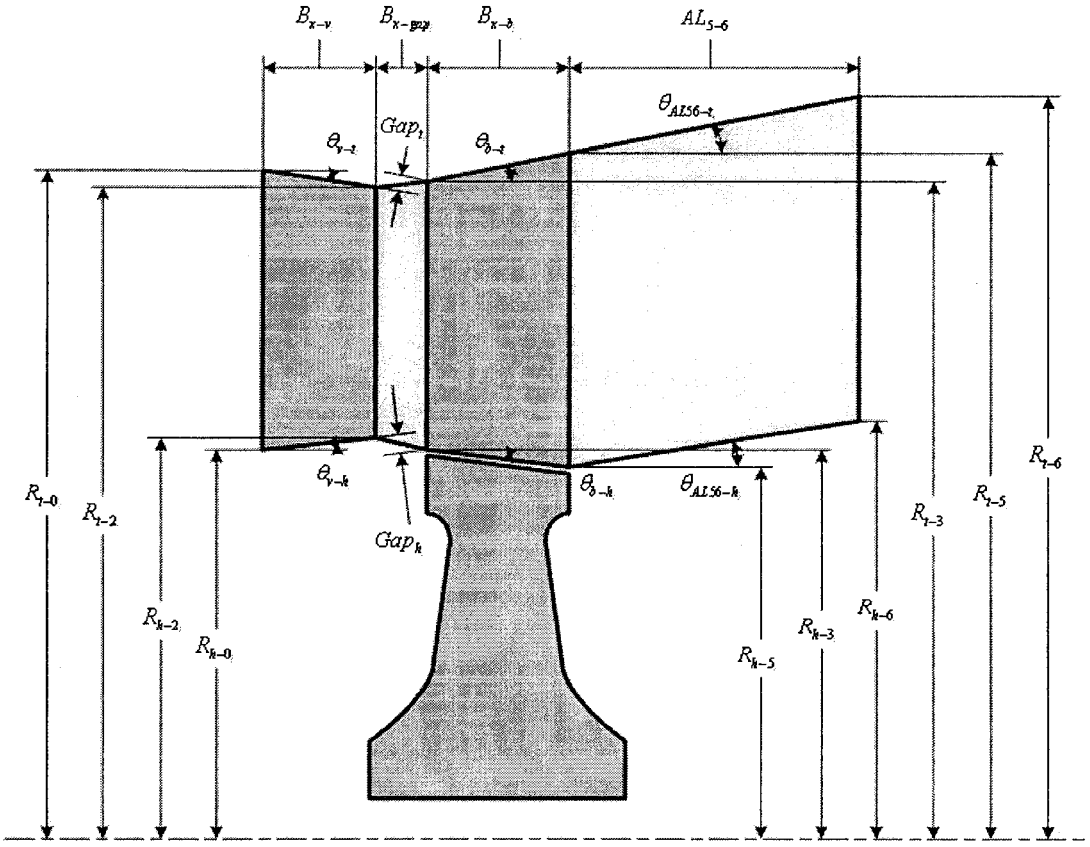

Figure I.1 Paramètres de la conduite de gaz

Si $i = 1$:

$$
\begin{aligned}
R_{h5_i} &= \frac{1}{2}\left(\frac{1^{st}Stage_{RimSpeed} \times 12 \times 60}{\pi|RPM_i|} + B_{x-b_i}\tan(\theta_{b-h_i})\right) \\
R_{t5_i} &= \sqrt{\frac{1^{st}Stage_{AN^2}}{\pi RPM_i^2} + R_{h5_i}^2} \\
R_{h3_i} &= R_{h5_i} - B_{x-b_i}\tan(\theta_{b-h_i}) \\
R_{t3_i} &= R_{t5_i} - B_{x-b_i}\tan(\theta_{b-t_i}) \\
R_{h2_i} &= R_{h3_i} - (B_{x-gap_i}\sin(\theta_{v-h_i}) - Gap_{h_i})/\cos(\theta_{v-h_i}) \\
R_{t2_i} &= R_{t3_i} - (B_{x-gap_i}\sin(\theta_{v-t_i}) - Gap_{t_i})/\cos(\theta_{v-t_i}) \\
R_{h0_i} &= R_{h2_i} - B_{x-v_i}\tan(\theta_{v-h_i}) \\
R_{t0_i} &= R_{t2_i} - B_{x-v_i}\tan(\theta_{v-t_i}) \\
R_{t6_i} &= R_{t5_i} + AL_{5-6_i}\tan(\theta_{AL56-t_i}) + Gap_{56-t} \\
R_{h6_i} &= R_{h5_i} + AL_{5-6_i}\tan(\theta_{AL56-h_i}) + Gap_{56-h}
\end{aligned}
\tag{I.2}
$$

Sinon :

$$
\begin{aligned}
R_{h0_i} &= R_{h6_{i-1}} \\
R_{t0_i} &= R_{t6_{i-1}} \\
R_{h2_i} &= R_{h0_i} + B_{x-v_i} \tan(\theta_{v-h_i}) \\
R_{t2_i} &= R_{t0_i} + B_{x-v_i} \tan(\theta_{v-t_i}) \\
R_{h3_i} &= R_{h2_i} + (B_{x-gap_i} \sin(\theta_{v-h_i}) - Gap_{h_i}) / \cos(\theta_{v-h_i}) \\
R_{t3_i} &= R_{t2_i} + (B_{x-gap_i} \sin(\theta_{v-t_i}) + Gap_{t_i}) / \cos(\theta_{v-t_i}) \\
R_{h5_i} &= R_{h3_i} + B_{x-b_i} \tan(\theta_{b-h_i}) \\
R_{t5_i} &= R_{t3_i} + B_{x-b_i} \tan(\theta_{b-t_i}) \\
R_{t6_i} &= R_{t5_i} + AL_{5-6_i} \tan(\theta_{AL56-t_i}) + Gap_{56-t} \\
R_{h6_i} &= R_{h5_i} + AL_{5-6_i} \tan(\theta_{AL56-h_i}) + Gap_{56-h}
\end{aligned}
\tag{I.3}
$$

où

$$
\begin{aligned}
1^{st}Stage_{RimSpeed} &= \frac{R_{h3_1} + R_{h5_1}}{2} \frac{\pi RPM_1}{12 \times 60} = (2R_{h5_1} - B_{x-b_1} \tan(\theta_{b-h_1})) \frac{\pi RPM_1}{12 \times 60} \\
1^{st}Stage_{AN^2} &= \pi \left(R_{t5_1}^2 - R_{h5_1}^2\right) RPM_1^2
\end{aligned}
\tag{I.4}
$$

On paramétrise les *Gaps* de cette façon parce qu'on veut toujours une expansion du fluide entre les aubes et les ailettes. Or, en paramétrisant les *Gaps* tel qu'indiqué à la figure I.1, il suffit ensuite de les contraindre à être positifs et on a ainsi une expansion assurée du fluide.

Le tableau I.1 présente une description de l'ensemble des variables utilisées pour la paramétrisation de la conduite de gaz.

Tableau I.1 Description des variables pour la conduite de gaz

Variables	Description	Type	Unités
θ_{v-h_i}	Angle de l'aube à l'emplanture	entrée	rad.
θ_{v-t_i}	Angle de l'aube à l'extrémité	entrée	rad.
θ_{b-h_i}	Angle de l'ailette à l'emplanture	entrée	rad.
θ_{b-t_i}	Angle de l'ailette à l'extrémité	entrée	rad.
θ_{AL56-t_i}	Angle du conduite inter-turbine à l'emplanture	entrée	rad.
θ_{AL56-h_i}	Angle du conduite inter-turbine à l'extrémité	entrée	rad.
Gap_{56-t}	Saut à la fin du conduite inter-turbine à l'emplanture	entrée	cm
Gap_{56-h}	Saut à la fin du conduite inter-turbine à l'extrémité	entrée	cm
Gap_{t_i}	Saut entre l'aube et l'ailette à l'emplanture	entrée	cm
Gap_{h_i}	Saut entre l'aube et l'ailette à l'extrémité	entrée	cm
B_{x-v_i}	Corde axiale des aubes	entrée	cm
B_{x-b_i}	Corde axiale des ailettes	entrée	cm
AL_{5-6_i}	Longueur axiale du conduite inter-turbine	entrée	cm
$1^{st}Stage_{AN^2}$	AN^2 du premier étage	entrée	cm.^{2}rpm^2
$1^{st}Stage_{RimSpeed}$	« Rim Speed » du premier étage	entrée	cm./s
NOV_i	Nombre d'aubes	entrée	
NOB_i	Nombre d'ailettes	entrée	

continue à la page suivante

suite de la page précédente

Variables	Description	Type	Unités
RPM_i	Révolution par minute	entrée	tr/min
$\Delta Reaction_i$	$Reaction_{optimale} - Reaction$	entrée	adim.
Δh_i	Variation d'enthalpie	couplée	kW/kg
W_0 (W_4)	Débit de gaz à l'entrée de la turbine	entrée	kg/s
T_0 (T_4)	Température à l'entrée de la turbine	couplée	K
p_0 (p_4)	Pression à l'entrée de la turbine	couplée	kPa
f/a	Débit de carburant / Débit d'air	couplée	adim.

I.3 Ailettes

Dans le programme RotorDesigner (RD), les ailettes non refroidies sont divisées en trois parties. L'aire à chacune des extrémités de ces parties doit être définie dans le fichier d'entrée. Il y a donc quatre aires à définir par ailette. De plus, une ailette peut contenir une cavité. Dans RD, il faut définir l'aire de cette cavité pour différentes sections. Finalement, on peut décider de refroidir ou non une ailette.

Lors de ce projet, les cavités ont été exclues. Les autres variables définies au tableau I.2 sont calculées à partir de la sortie de RD ou sont fixées et donc deviennent des paramètres. L'ailette est ainsi complètement paramétrisée mais le prix à payer est qu'il faut exécuter RD deux fois pour obtenir les résultats, car pour calculer l'aire des ailettes, il faut connaître le chargement. L'équation I.5 montre les détails des calculs

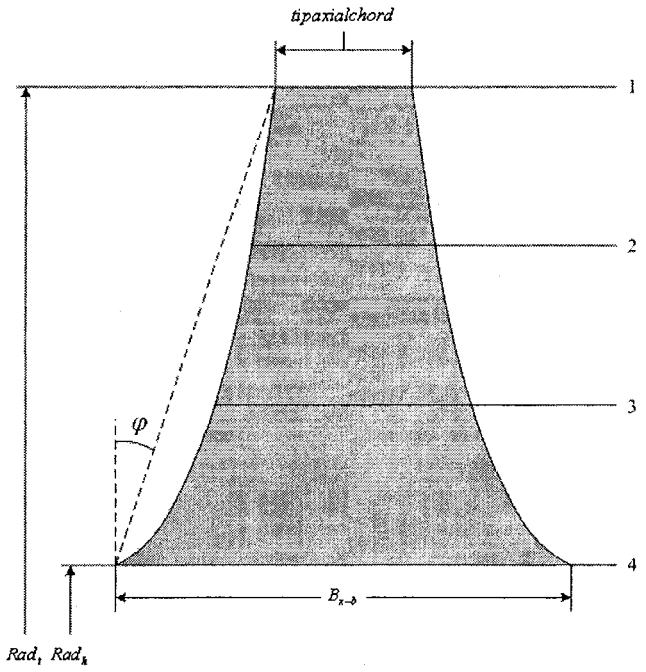

Figure I.2 Paramètres de l'ailette

$$
\begin{aligned}
areaatrad_1 &= \max\left\{\frac{Traction_{épaulement}}{Contrainte_{max_{a1}}}, \tau B_{x-b}\right\} \\
areaatrad_4 &= \max\left\{\frac{Traction_{épaulement} + Traction_{ailettes}}{Contrainte_{max_{a4}}}, \zeta areaatrad_1\right\} \\
areaatrad_2 &= areaatrad_1 + \phi(areaatrad_4 - areaatrad_1) \\
areaatrad_3 &= areaatrad_1 + \omega(areaatrad_4 - areaatrad_1) \\
tipaxialchord &= B_{x-b} - 2(Rad_t - Rad_h)\tan(\varphi)
\end{aligned}
\tag{I.5}
$$

où

τ est un facteur qui donne une aire minimale de l'ailette à l'extrémité

ζ est un facteur basé sur la dynamique de l'ailette

ϕ et ω sont des pourcentages de la différence entre $areaatrad_4$ et $areaatrad_1$

φ est un angle (voir la figure I.2)

$$Rad_h = \frac{R_{h3} + R_{h5}}{2}, \qquad Rad_t = \frac{R_{t3} + R_{t5}}{2}$$

Le tableau I.2 présente l'ensemble des variables pour les ailettes.

Tableau I.2 Description des variables pour l'ailette

Variables	Description	Type	Unités
$areaatrad_1$	aire de l'ailette à l'extrémité	entrée	cm^2
$areaatrad_2$	aire de l'ailette au 2/3 de sa longueur radiale	entrée	cm^2
$areaatrad_3$	aire de l'ailette au 1/3 de sa longueur radiale	entrée	cm^2
$areaatrad_4$	aire de l'ailette à l'emplanture	entrée	cm^2
tipaxialchord	corde axiale à l'extrémité de l'ailette	entrée	cm
topsection	numéro de la section où débute la cavité	entrée	
midsection	numéro de la section au milieu la cavité	entrée	
endsection	numéro de la section à la fin la cavité	entrée	
topcavityarea	aire de la section où débute la cavité	entrée	cm^2
midcavityarea	aire de la section au milieu la cavité	entrée	cm^2
endcavityarea	aire de la section à la fin la cavité	entrée	cm^2
matdepreciation	dégradation du matériel de l'ailette	entrée	K
shroudbendstress	contrainte en flexion dans l'épaulement	entrée	kPa
shroudscalefactor	facteur d'échelle dans l'épaulement	entrée	kPa

I.4 Plateforme

La plateforme utilisée par RD est assez rudimentaire. On calcule les quelques variables de la manière suivante :

$$\begin{aligned} platformtickness &= (Rad_h - R_{h5}) + \Delta_{thickness} \\ newaxialchord &= B_{x-b} \end{aligned} \tag{I.6}$$

où

$\Delta_{thickness}$ est basé sur l'angle de brochage des attaches

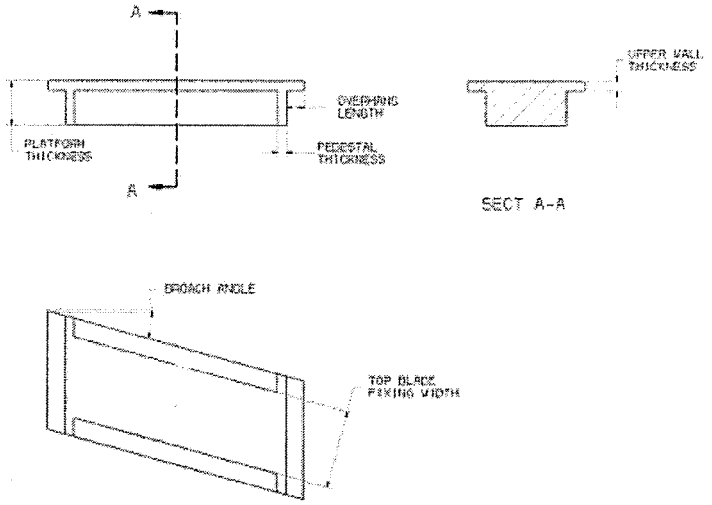

Figure I.3 Paramètres de la plateforme

Le tableau I.3 présente l'ensemble des variables pour la plateforme.

Tableau I.3 Description des variables pour la plateforme

Variables	Description	Type	Unités
upwallthickness	épaisseur de la paroi du haut	entrée	cm
overhanglength		entrée	cm

continue à la page suivante

suite de la page précédente

Variables	Description	Type	Unités
pedestalthickness		entrée	cm
platformtickness	épaisseur de la plateforme	entrée	cm
newaxialchord	Corde axiale de l'ailette refroidie	entrée	cm

I.5 Attaches

Les attaches constituent la partie la plus complexe car un grand nombre de variables est nécessaire pour les paramétriser. Il n'est pas évident de définir des règles permettant de réduire le nombre de variables. En fait, seul le *flatlength* est calculé à partir du chargement et toutes les autres dimensions sont des variables tel que montré à l'équation I.7

$$flatlength_j = v\frac{F_{total_i}}{2B_{x-b_i} contrainte_{max} nb_{lobes_i}}, \qquad j = 1, \ldots, nb_{lobes_i} \tag{I.7}$$

où

v est un facteur qui dépend du type de turbine (HPT, PT, FAN)

Le tableau I.4 présente l'ensemble des variables pour les attaches.

Tableau I.4 Description des variables pour les attaches

Variables	Description	Type	Unités
caarea	Aire de l'air de refroidissement	entrée	cm^2
$wedge_{angle}$	Angle du wedge	entrée	°
$drload_{angle}$	Angle de chargement dessiné	entrée	°
$drunload_{angle}$	Angle de relâchement dessiné	entrée	°
$topblade_{angle}$	Angle au début du premier lobe	entrée	°

continue à la page suivante

suite de la page précédente

Variables	Description	Type	Unités
$topneck_{width}$	Largeur du premier creux	entrée	cm
$rlobe_i$	Rayon du lobe i	entrée	cm
$rneck_i$	Rayon dans le creux du lobe i	entrée	cm
$flatlength_i$	Longueur plane du lobe i	entrée	cm
$offset_i$	Décallage du lobe i	entrée	cm
$rimwidth$	Largeur du rim	entrée	cm
$broach_{angle}$	Angle de brochage	entrée	°
$fixing_{r2}$			
$bottomlobe_{angle}$	Angle du dernier lobe	entrée	°
$bottomlobe_{radius}$	Rayon du dernier lobe	entrée	cm
$rivet_{radius}$	Rayon du rivet	entrée	cm
$rivet_{a1}$			

I.6 Disque

Dans le logiciel RD, le disque doit obligatoirement être symétrique, ce qui vient diviser le nombre de variables par presque deux. Puisqu'un disque symétrique ne correspond pas à la réalité, il ne sert à rien d'aller chercher un grand niveau de détail dans son design. Pour cette raison, le disque est en grande partie paramétrisé, donc seulement quelques variables suffisent à le définir. Il n'en résulte pas pour autant un design dépourvu de sens car ce design représente en grande partie ce que les ingénieurs font pour obtenir un design de départ.

On calcule les variables comme suit :

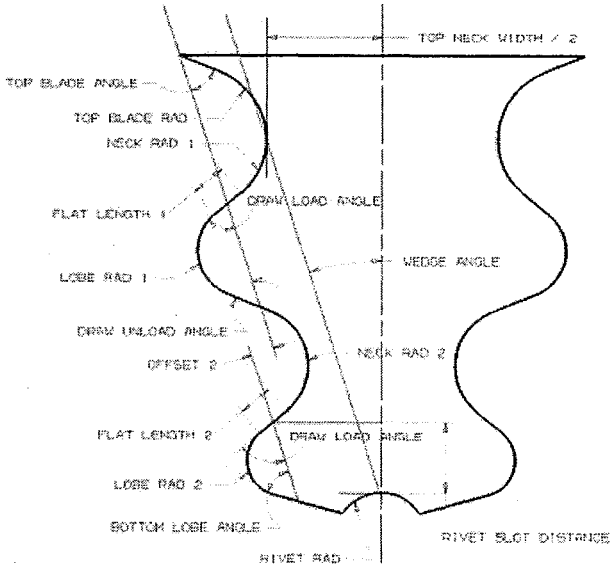

Figure I.4 Paramètres des attaches

$$
\begin{aligned}
discboreradius &= \kappa liverimrad \\
discborewidth &= \xi B_{x-b} \\
discboreheight &= liverimrad - discboreradius
\end{aligned}
\tag{I.8}
$$

où

κ est un % du $liverimrad$

ξ est un % de B_{x-b}

Le tableau I.5 présente l'ensemble des variables pour le disque.

Tableau I.5 Description des variables pour le disque

Variables	Description	Type	Unités
thickness	voir le symbole T sur la figure I.5	entrée	cm
highneckheight	% de la hauteur du disque	entrée	adim.
highneckwidth	% du *rimwidth*	entrée	adim.
highneckrad	rayon dans la partie supérieure du disque	entrée	cm

continue à la page suivante

suite de la page précédente

Variables	Description	Type	Unités
lowneckheight	% de la hauteur du disque	entrée	adim.
lowneckwidth	% de *discborewidth*	entrée	adim.
lowneckrad	rayon dans la partie inférieure du disque	entrée	cm
discboreradius	rayon interne du disque	entrée	cm
discborewidth	largeur du disque à son rayon interne	entrée	cm
discboreheight	% de la hauteur du disque	entrée	cm
rimwidth	% de B_{x-b}	entrée	cm
MUF	facteur d'utilisation du matériel du disque	entrée	adim.

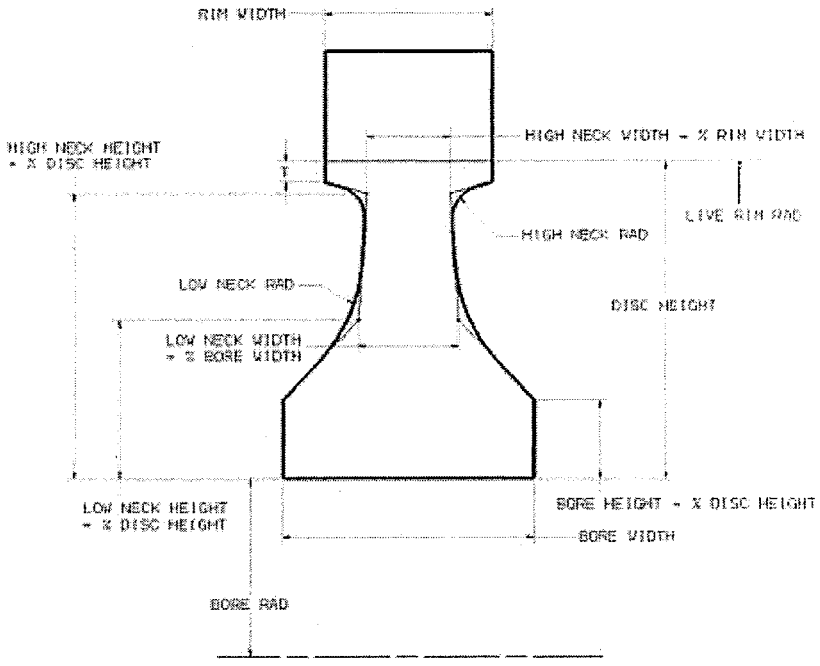

Figure I.5 Paramètres du disque

ANNEXE II

MÉTHODE À GRADIENT - NLPQL

Voici les détails mathématiques de l'algorithme de programmation non-linéaire (NLP, Non-Linear Programming) NLPQL tel que définis par Schittkowski [19]. On assume que les bornes sur les variables sont traitées comme des contraintes d'inégalités afin de simplifier l'écriture.

Le problème se formule de la manière suivante :

$$\begin{array}{lll} & \min f(x) & \\ x \in I\!R^n & g_j(x) = 0, & j = 1, \ldots, m_e \\ & g_j(x) \geq 0, & j = m_{e+1}, \ldots, m \end{array} \tag{II.1}$$

Le Lagrangien est donc :

$$L(x, u) = f(x) - \sum_{j=1}^{m} u_i g_j(x) \tag{II.2}$$

où les u_i sont les multiplicateurs de Karush-Kuhn-Tucker (KKT).

Pour formuler le problème quadratique, on part d'une approximation de la solution $x_k \in I\!R^n$, d'une approximation des multiplicateurs de KKT $v_k \in I\!R^m$ et d'une approximation du Hessien du Lagrangien de f $B_k \in I\!R^{nxn}$.

$$\begin{array}{lll} & \min \frac{1}{2} d^T B_k d + \nabla f(x_k)^T d & \\ x \in I\!R^n & \nabla g_j(x_k)^T + g_j(x_k) = 0, & j = 1, \ldots, m_e \\ & \nabla g_j(x_k)^T + g_j(x_k) \geq 0, & j = m_e, \ldots, m \end{array} \tag{II.3}$$

On peut ainsi résoudre facilement ce sous-problème avec un algorithme de type QP (Quadratic Programming), en occurrence un code basé sur le dual du problème et nommé QL, pour obtenir d_k et u_k. On obtient alors les valeurs pour l'itération suivante :

$$\begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ v_k \end{pmatrix} + \alpha_k \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \tag{II.4}$$

où $\alpha_k \in [0, 1]$ est le pas.

On cherche α_k tel que la fonction coût suivante diminue suffisamment :

$$\phi_r(x, v) = \psi_r \left(\begin{pmatrix} x \\ v \end{pmatrix} + \begin{pmatrix} d \\ u - v \end{pmatrix} \right) \tag{II.5}$$

où ψ_r est la fonction de pénalité suivante :

$$\psi_r(x, v) = f(x) - \sum_{j \in J} (v_j g_j(x) - \frac{1}{2} r_j g_j(x)^2) - \frac{1}{2} \sum_{j \in K} \frac{v_j^2}{r_j} \tag{II.6}$$

où $J = \{1, \ldots, m_e\} \cup \{j\ , m_e < j \leq m, g_j(x) \leq v_j r_j\}$ et $K = \{1, \ldots, m\} \backslash J$

Le paramètre de pénalité $r_j, j = 1, \ldots, m$ est choisi afin d'assurer une diminution de la fonction mérite :

$$\phi'_{r_k} = \nabla \psi_{r_k}(x_k, v_k)^T \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \tag{II.7}$$

Le pas α_k est choisi de manière à satisfaire la condition d'Armijo :

$$\phi_r(\sigma \beta^i) \leq \phi_r(0) + \sigma \beta^i \mu \phi'_r(0) \tag{II.8}$$

où $0 \leq \mu \leq 0.5$, $0 \leq \beta \leq 1$ et $0 \leq \sigma \leq 1$

La procédure consiste à partir avec $i = 0$ et de l'augmenter jusqu'à ce que la condition d'Armijo soit satisfaite. On obtient ainsi $\alpha_k = \sigma\beta^{i_k}$.

En résumé, on trouve une direction locale de descente à partir d'un point par la résolution d'un problème QP approximant de façon quadratique la solution localement. On cherche ensuite une distance de descente dans cette direction de manière à minimiser une certaine fonction coût tenant compte de la violation des contraintes. On trouve ainsi un nouveau point par l'équation II.3 et on recommence jusqu'à convergence.

ANNEXE III

ANALYSE DE SENSIBILITÉ

Pour utiliser les méthodes à gradients, il est nécessaire de définir un pas de différenciation commun à toutes les variables. Pour ce faire, une étude de sensibilité a été effectuée pour SOAPP et P1242. L'étude de sensibilité de RD n'a pas été faite et on a supposé que le pas trouvé pour SOAPP et P1242 conviendrait à RD. On entend par pas convenable, un pas permettant à la fois de s'affranchir du bruit numérique des programmes d'analyses tout en fournissant des gradients précis. La notion de précision est importante en ingénierie et dans le cas présent, une précision relative de 0.1% est suffisante. On cherche donc ici un pas relatif de différenciation idéalement inférieur à 10^{-3} et évitant le bruit numérique.

En pratique, on déterminera ensuite l'intervalle de pas de différenciation convenant au trois programmes à l'aide de tests d'optimisations. L'intervalle de départ pour ces tests sera celui trouvé par la présente étude de sensibilité. On présente donc dans cette annexe les résultats de chaque étude de sensibilité.

III.1 Sensibilité de SOAPP

Pour SOAPP, on vérifie la sensibilité du *sfc* envers les variables qui seront utilisées pour l'optimisation. La figure III.1 donne les graphes de la sensibilité des gradients de quelques variables typiques en fonction du logarithme du pas de différenciation. Comme on l'observe, il n'est pas évident de trouver un pas de différenciation qui plaît à tous tout en étant suffisamment petit pour permettre la convergence d'une optimisation. Mis à part le $BLWA7$, un pas entre 4×10^{-3} et 10^{-2} permet d'éviter le bruit. On a déterminé cette intervalle visuellement en ce basant sur les graphes de la

figure III.1. Pour plusieurs de ces variables, l'intervalle choisi est située dans le bruit numérique. Ceci n'est pas vraiment un problème pour deux raisons :

1. L'amplitude du bruit est très petite pour ces variables ;
2. De plus, il n'y a pas de changement de signe du gradient dû au bruit numérique pour ces variables.

III.2 Sensibilité de P1242

Pour P1242, on vérifie la sensibilité de la somme des rendements envers les variables qui seront utilisées pour l'optimisation. La figure III.2 donne les graphes de la sensibilité des gradients de quelques variables typiques en fonction du logarithme du pas de différenciation. Comme on l'observe, il n'est encore une fois pas évident de trouver un pas de différenciation qui plaît à tous tout en étant suffisamment petit pour permettre la convergence d'une optimisation. Un pas entre 10^{-3} et 10^{-2} semble un bon compromis.

En résumé, les études de sensibilités effectuées pour SOAPP et P1242 semblent démontrer qu'un pas de différenciation situé approximativement dans l'intervalle $[4 \times 10^{-3}, 10^{-2}]$ permettrait de limiter les problèmes liés au bruit tout en fournissant des gradients d'une précision acceptable compte tenu de la précision des programmes d'analyses.

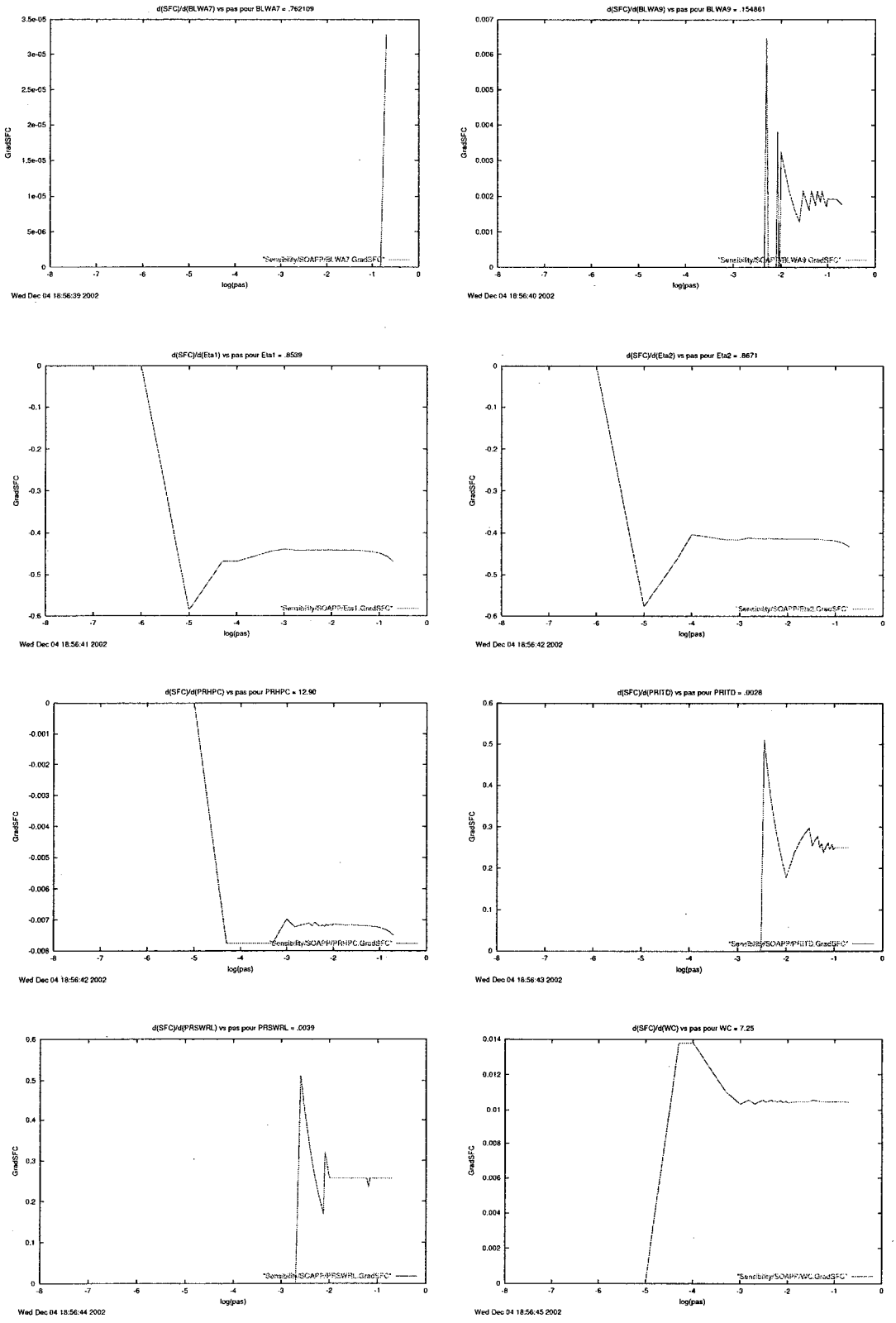

Figure III.1 Résultats de l'étude de sensibilité de SOAPP pour quelques variables

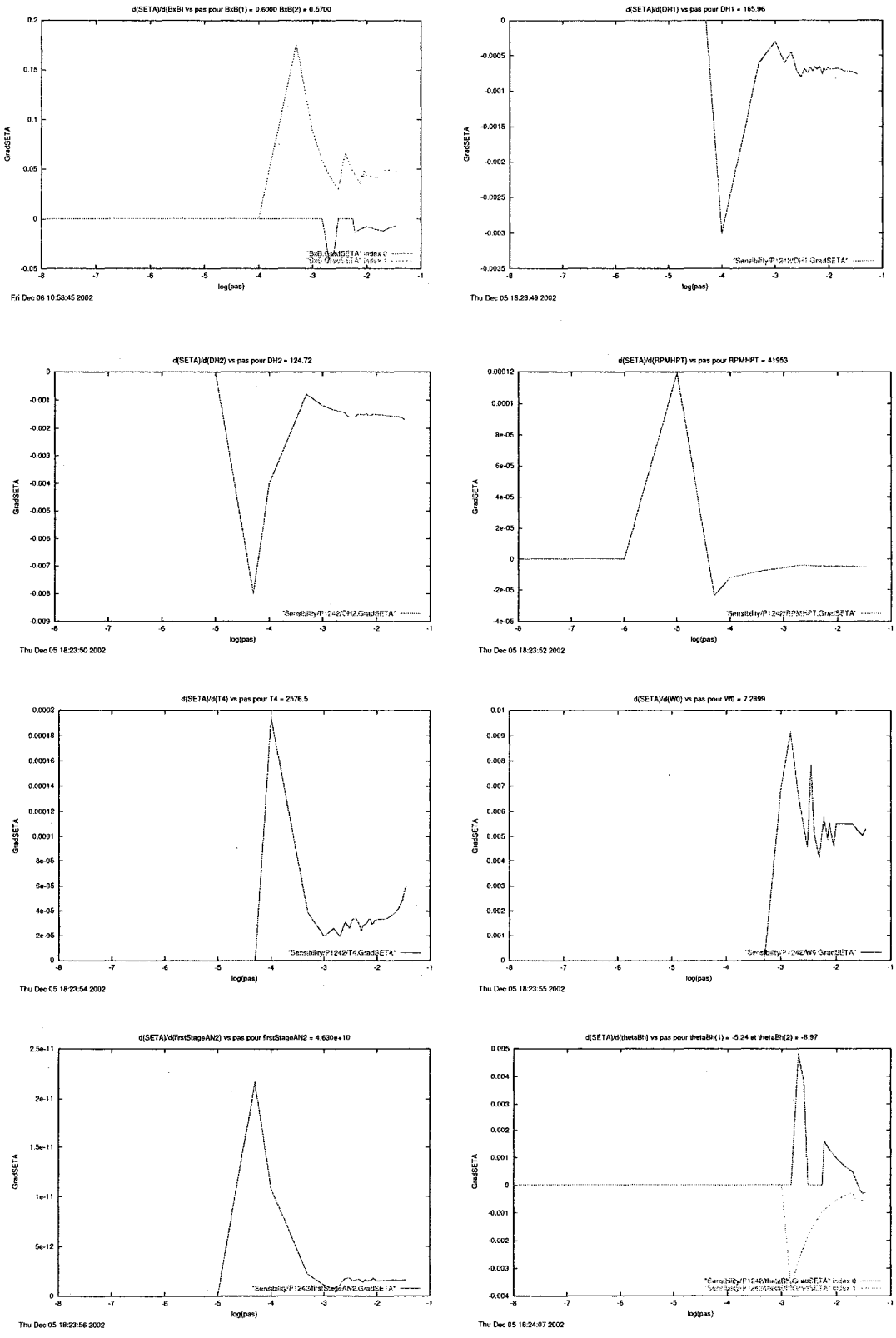

Figure III.2 Résultats de l'étude de sensibilité de P1242 pour quelques variables